Universidad Nacional Mayor de San Marcos
Universidad del Perú. Decana de América
Facultad de Ingeniería de Sistemas e Informática
Escuela Académico Profesional de Ingeniería de Sistemas

"Implementación del Sistema de Gestión de Fondos Concursales para gestionar las subvenciones otorgadas por el CONCYTEC utilizando el framework JSF"

TESINA

Para optar el Título Profesional de Ingeniera de Sistemas

AUTOR

Milagros LAMAS RODRÍGUEZ

ASESOR

Alecxy DÍAZ SANDOVAL

Lima, Perú

2006

Referencia bibliográfica

Lamas, M. (2006). *Implementación del Sistema de Gestión de Fondos Concursales para gestionar las subvenciones otorgadas por el CONCYTEC utilizando el framework JSF*. Tesina para optar el título profesional de Ingeniero de Sistemas. Escuela Académica Profesional de Ingeniería de Sistemas, Facultad de Ingeniería de Sistemas e Informática, Universidad Nacional Mayor de San Marcos, Lima, Perú.

DEDICATORIA: Dedico este trabajo a mi *Alma Mater*, la Universidad Nacional Mayor de San Marcos, por que en sus aulas adquirí los conocimientos que hoy me permiten desarrollarme profesionalmente.

AGRADECIMIENTOS

Agradezco especialmente a mis padres que me dieron las cosas más preciadas: su amor, su apoyo y la educación, en una universidad tan prestigiosa como San Marcos.

También agradezco a cada uno de mis profesores a mi paso por la Universidad Mayor de San Marcos, ya que en mis cinco años de estudios supieron verter su conocimiento y despertar la inquietud de investigar. Gracias a sus innumerables trabajos, los que hicieron que me esforzara cada vez más.

INDICE

RESUMEN

IMPLEMENTACION DEL SISTEMA DE GESTION DE FONDOS CONCURSALES PARA GESTIONAR LAS SUBVENCIONES OTORGADAS POR EL CONCYTEC UTILIZANDO EL FRAMEWORK JSF

Milagros Lamas Rodriguez

Junio-2006

Asesor : Alecxy Diaz Sandoval

Grado : Tesina

El Consejo Nacional de Ciencia, Tecnología e Innovación Tecnológica (CONCYTEC), tiene como uno de sus objetivos motivar e incentivar la vocación por las ciencias, promover el desarrollo de las habilidades y capacidad creativa de los recursos humanos, así como la difusión de los conocimientos y avances científicos y tecnológicos a nivel nacional, para el cumplimiento de dicho objetivo otorga subvenciones periódicamente bajo diferentes categorías: Becas, Tesis, Premios, Eventos, Publicaciones y Proyectos.

Los principales procesos durante una convocatoria son Postulación, Evaluación, Contratos y Seguimiento. Cabe indicar que es necesario realizar un seguimiento efectivo de las subvenciones otorgadas, ya que este resulta ser un proceso crítico del negocio. Para ello el uso de sistemas informáticos son de gran ayuda, ya que permiten automatizar algunos procesos del negocio y de esta manera incrementar la productividad.

En el presente trabajo se analiza la situación actual de la gestión de las subvenciones, y se propone la construcción de un sistema informático web para realizar dicha gestión, el Sistema de Gestión de Fondos Concursales, el cual permitirá dar soporte a cada uno de los procesos involucrados. Además se analizan los fundamentos en los que se basa el marco de trabajo JSF, API Java estándar para construir interfaces de usuario en aplicaciones Web. Éste marco de trabajo se enfoca en la capa de la Vista de la arquitectura Modelo-Vista-Controlador, aunque proporciona suficiente capacidad de control para escribir aplicaciones moderadamente complejas utilizando un único API.

Palabras claves: Sistema de información web, Aplicaciones web, tecnologías de desarrollo, programación orientada a objetos, diseño en capas, contenido dinámico, patrones de diseño.

ABSTRACT

IMPLEMENTATION OF THE SYSTEM OF MANAGEMENT OF SUBVENTIONS GRANTED BY THE CONCYTEC BEING USED FRAMEWORK JSF

Milagros Lamas Rodriguez

June-2006

Adviser : Alecxy Diaz Sandoval

Degree : Tesina

The National Council of Science, Technology and Technological Innovation (CONCYTEC), must like one of its objectives motivate and stimulate the vocation by sciences, to promote the scientific and technological development of the abilities and creative capacity of the human resources, as well as diffusion of the knowledge and advances at national level, for the fulfillment of this objective grants subventions under different categories periodically: Scholarships, Thesis, Prizes, Events, Publications and Projects.

The main processes during a call are Postulation, Evaluation, Contracts and Pursuit. It is possible to indicate that it is necessary to make an effective

pursuit of the granted subventions, since this turns out to be a critical process of the business. For it the use of computer science systems is helpful, since they allow to automate some processes of the business and this way of increasing the productivity.

In the present work the present situation of the management of the subventions is analyzed, and the construction of a computer science system sets out Web to make this management, the System of Management of Bottoms, which will allow to give support to each one of the involved processes. In addition are analyzed the foundations on which the framework JSF is based, API standard Java to construct to interfaces of user in applications Web. This framework focuses in the layer of the Vista of the architecture Model-Vista-Controller, although it provides sufficient capacity of control to write moderately complex applications using an only API.

Key words: Information system Web, Applications Web, technologies of development, object-oriented programming, design in layers, dynamic content, patterns of design.

INTRODUCCION

El Consejo Nacional de Ciencia, Tecnología e Innovación Tecnológica (CONCYTEC), en su papel de ente promotor de la investigación y uso de la tecnología en el Perú, otorga subvenciones bajo diferentes categorías, las que es necesario gestionarlas de manera efectiva, para ello requiere una solución informática integral.

Actualmente existe un sistema de escritorio: *"Sistema de Subvenciones"* para la gestión de las subvenciones, el cual no está completamente terminado ya que no cubre la etapa del seguimiento a los subvencionados, la que es una etapa crítica del proceso de entrega de subvenciones. La solución propuesta consiste en un sistema de información web desarrollado utilizando software libre utilizando JSF como marco de trabajo.

Una ventaja de la implementación de una solución web es que permitirá una mayor y más fácil interacción de los involucrados en el proceso de subvenciones, ya que los postulantes a una subvención podrán ingresar su expediente vía web, así como consultar el estado del proceso. Además se puede ampliar el número de evaluadores participantes. Actualmente los

evaluadores sólo pueden ser personas que físicamente puedan ir al CONCYTEC para evaluar los expedientes, sin embargo existen profesionales que podrían participar como evaluadores pero se encuentran fuera de Lima, en la solución propuesta, pueden participar evaluadores que residan fuera de Lima ya que la evaluación la podrán realizar vía web.

Por otro lado se ha escogido el marco de trabajo JSF por las ventajas que representa usarlo, ya que permite un desarrollo de aplicaciones web más rápido debido a que se basa en la utilización de componentes, similar o lo que se puede desarrollar con asp .NET.

Con la solución propuesta se busca lograr una mayor rapidez y eficacia en la comunicación entre todas las dependencias que intervienen en el proceso de entrega de subvenciones, y así obtener información confiable para la toma de decisiones.

CAPITULO I

1.- PLANTEAMIENTO DEL PROBLEMA

1.1 Antecedentes y formulación del problema

El CONCYTEC (Consejo Nacional de Ciencia, Tecnología e Innovación Tecnológica) desde hace varios años viene apoyando a la investigación de diversas formas, tal es así que otorga becas para estudios de postgrado en universidades nacionales e internacionales, así como subvenciones a proyectos de investigación, publicaciones, eventos, tesis de postgrado y premios a investigadores a manera de estimular la investigación en el Perú.

Para ello el CONCYTEC dispone de unidades o dependencias que gestionan estas becas: la Dirección General de Apoyo a la Investigación (DGAI) y la Dirección General de Formación en Ciencia y Tecnología (DGFCYT). De la misma forma el manejo de información para registrar, procesar y generar reportes ha sido un trabajo continuo por parte de los responsables de dichas unidades.

Actualmente existe un sistema de escritorio: *"Sistema de Subvenciones",* el que se utiliza para la gestión de convocatorias, postulaciones, evaluación y

generación de contratos; pero no satisface las expectativas de los usuarios ni de los clientes (subvencionados).

1.2 Justificación e importancia.

Un porcentaje significativo del presupuesto del CONCYTEC se destina anualmente para otorgar estas subvenciones, el que fluctúa entre el 29% y 31%. En los reglamentos de dichas subvenciones se ha establecido que los subvencionados deben presentar rendiciones de lo recibido, el sistema actual no contempla la fase de seguimiento a los subvencionados, por lo que existen 4 millones de soles de los que no se tiene la certeza si fueron rendidos o no.

1.3 Definición del problema.

Debido a la falta de automatización de algunos procesos correspondientes a la etapa de seguimiento de una subvención, se ha optado por realizar el registro de las rendiciones de los subvencionados en hojas de cálculo Excel, las que no poseen ninguna integración con el *"Sistema de Subvenciones"*, esto genera grandes dificultades en obtener información confiable y disponible para tomar decisiones, acrecentando la existencia de información no confiable. Como resultado de esto los más afectados son los subvencionados.

Asimismo en cada convocatoria se observan largas colas los últimos días de cierre y la acumulación de gran cantidad de expedientes. Los interesados en estas subvenciones tienen grandes dificultades en especial los de provincia.

Otro inconveniente surge cuando en cada proceso de subvención se necesita tener la relación de deudores (subvencionados que no han rendido sus gastos o presentado los informes solicitados), pues el proceso de rendiciones se efectúa en la oficina de administración. Para solucionar esto se tienen que revisar los archivos (Hojas de cálculo excel) y confrontarlos con las rendiciones efectuadas en la DGAI (Dirección General de Apoyo a la Investigación). De la misma forma al término de la subvención se pierde el control de los informes y no se puede llevar un seguimiento efectivo.

1.4 Limitaciones y alcance.

El alcance de la investigación llevará a la implementación del "*Sistema de Gestión de Fondos Concursales"* SGFC utilizando el marco de trabajo JSF.

1.5 Variantes del problema.

Así como Concytec otorga subvenciones, existen otras instituciones que también lo hacen, dichas instituciones pueden requerir que una vez realizada la convocatoria, evaluación y selección de ganadores, se le haga el seguimiento a los subvencionados. El sistema debe ser lo suficientemente flexible y genérico en el que se pueda hacer seguimiento a subvenciones de algún otro fondo económico ajeno al de Concytec.

CAPITULO II

2.- OBJETIVOS

2.1 Objetivos generales.

- Implementar un sistema integral para la gestión de las subvenciones otorgadas por Concytec, de esta forma brindar un mejor servicio a los usuarios y ciudadanos; y tener información confiable y oportuna.
- Hacer un análisis de los fundamentos en que se basa el framework JSF, aplicarlos y así obtener resultados que nos muestren su verdadero alcance.

2.2 Objetivos específicos.

- Realizar un análisis del proceso de entrega de subvenciones por parte de Concytec, a fin de Identificar procesos críticos que ameriten mayor atención en el modelado del negocio.
- Realizar un análisis de la situación actual del sistema de subvenciones utilizado por Concytec para la gestión de subvenciones, identificando áreas criticas para su posterior corrección en la nueva implementación de dicho sistema.

- Implementar *el "Sistema de Gestión de Fondos Concursales" (*SGFC).

CAPITULO III

3.- MARCO TEORICO CONCEPTUAL

3.1 Antecedentes de la Investigación.

Existe poca referencia sobre el framework JSF en español, sin embargo se puede citar a algunos autores de bibliografia en inglés: Kito D. Mann con su obra JAVA SERVER FACES IN ACTION, brinda un gran aporte para los que desean conocer en forma práctica, qué es y cómo funciona el framework JSF.

Otros autores notables son David Geary y Cay Horstmann, quienes en su obra CORE JAVA SERVER FACES, nos explican detalladamente las bondades de este framework, acompañando las explicaciones con ejemplos prácticos. Ambos autores coinciden en señalar que este marco de trabajo está en pleno desarrollo, y le auguran muchos éxitos en su uso.

Esto se puede justificar por el soporte que vienen dando las compañias Oracle, IBM entre otras, ya que sus IDE's (Integrated Development Environment, Entorno de Desarrollo Integrado) vienen incorporando soporte JSF, tal es el caso de JDeveloper de Oracle, WebSphere Application

Developer de IBM, Java Studio Creador de Sun, Eclipse con el pluggin Myeclipse, Neatbeans, etc.

3.2 Bases teóricas.

3.2.1 ¿Por qué utilizar una aplicación web en lugar de una de escritorio?

Si duda cada tecnología resuelve un tipo de problema y presenta ventajas y desventajas, a la hora de decidir por una se debe contrastar las ventajas que ésta nos ofrecen con las ventajas que nos ofrecen las demás opciones, así como sus respectivas desventajas. A continuación se enumeran las ventajas y desventajas de las aplicaciones de escritorio y de las aplicaciones web, para que nos dé una idea de los motivos por los cuales se debería optar por una aplicación web en lugar de una de escritorio.

Aplicación de escritorio

Ventajas

- Respuesta inmediata a los eventos que realiza el usuario
- Vistosas interfaces, buena capacidad gráfica
- Múltiples posibilidades de interacción con el usuario
- Debido a que es más antiguo posee estándares ampliamente soportados lo que facilita el desarrollo de las aplicaciones

Desventajas

- Requiere de instalación en cada usuario
- Monousuario. Sólo el usuario frente al ordenador puede utilizar la aplicación en un momento dado.

- Sólo pueden funcionar bajo el sistema para el que fueron diseñadas.

Aplicación Web

Ventajas

- **Ubicuidad.** Se accede a la aplicación desde cualquier equipo informático (computadora personal, PDA) conectado a Internet, con independencia de su situación geográfica. Así, es posible controlar en todo momento la situación y contenidos de la web, tanto desde el propio local como desde casa, o incluso desde el extranjero.
- **Multiusuario.** A diferencia de las aplicaciones de escritorio, donde sólo el usuario frente a la computadora puede utilizar la aplicación, en las aplicaciones web puede haber varios usuarios conectados al sistema simultáneamente, cada uno a través de un ordenador distinto y en unas ubicaciones geográficas separadas (empresas con varios almacenes, por ejemplo), y todos utilizar la aplicación con normalidad.
- **Independencia de software**. Para acceder a la aplicación sólo es requisito un navegador web estándar, sin necesidad de instalar en cada equipo ningún otro programa específico. Debido a estos bajos requerimientos, el software puede utilizarse incluso desde computadoras obsoletas y poco potentes.
- **Seguridad**. Al albergarse en un servidor remoto, el funcionamiento de la aplicación y los valiosos datos que contiene

son totalmente independientes de la computadora utilizada para la gestión. Así, la normal operación de la aplicación es inmune a una avería de hardware, virus informáticos, o cualquier otro problema local. En caso de necesidad, basta con retomar la conexión al servidor desde cualquier otra computadora personal o portátil.

- **Multiplataforma e interoperabilidad**. A diferencia de las aplicaciones de escritorio, que sólo pueden funcionar bajo el sistema para el que fueron diseñadas, las aplicaciones web son multiplataforma por diseño. Esto significa que podrá conectar con el software desde cualquier versión de Windows -presente o futura-, o incluso otros sistemas operativos como GNU/Linux, Solaris, Symbian (teléfonos móviles GPRS).
- **Actualización.** Las aplicaciones basadas en web están siempre actualizadas con el último lanzamiento sin requerir que el usuario tome acciones pro-activas, y sin necesitar llamar la atención del usuario o interferir con sus hábitos de trabajo con la esperanza de que va a iniciar nuevas descargas y procedimientos de instalación.
- **Menos requerimientos de memoria.** Las aplicaciones basadas en web tienen muchas más razonables demandas de memoria RAM de parte del usuario final que los programas instalados localmente. Al residir y correr en los servidores del proveedor, a esas aplicaciones basadas en web usa en muchos casos la memoria de las computadoras que ellos corren, dejando más

espacio para correr múltiples aplicaciones del mismo tiempo sin incurrir en frustrantes deterioros en el rendimiento.

- **Menos Bugs (**Errores en la codificación**).** Las aplicaciones basadas en web deberían ser menos propensas a colgarse y crear problemas técnicos debido a software o conflictos de hardware con otras aplicaciones existentes, protocolos o software personal interno. Con aplicaciones basadas en web, todos utilizan la misma versión, y todos los bugs pueden ser corregido tan pronto como son descubiertos. Esta es la razón por la cual las aplicaciones basadas en web deberían tener mucho menos bugs que el software de escritorio descargable tradicional.
- **Precio.** Las aplicaciones basadas en web no requieren la infraestructura de distribución, soporte técnico y marketing requerido por el software descargable tradicional. Esto permite que las aplicaciones online cuesten una fracción de sus contrapartes descargables si no totalmente gratuitas, mientras que ofrecen componentes adicionales y servicios premium como una opción.
- **Los datos también van online.** Por supuesto con el desplazamiento de las aplicaciones locales a aquellas basadas en web también los datos que creamos y accedemos van a necesitar experimentar profundos cambios. A nadie le gusta no poder acceder a su propio e-mail cuando está de viaje, o poder recuperar un documento particular cuando se conecta desde un ciber café a 15.000 kilómetros de su oficina.

- **Los datos son más seguros.** Si bien la ruptura de discos no va a desaparecer, es probable que los usuarios escuchen mucho menos del tema. A medida que las compañías se haga cargo del almacenamiento de los datos del usuario, granjas de almacenamiento de datos redundantes, altamente fiables, serán la norma más que la excepción, y los usuarios van a tener mucho menos riesgo de perder sus datos debido a una ruptura de disco impredecible o a un virus de la computadora. Las compañías que provee aplicaciones basadas en web van a brindar amplios servicios de resguardo de datos ya sea como una parte integral del servicio básico o cobrando por ello. Usted puede imaginar que si una compañía comercial pierde los datos de la gente será puesta de rodillas (financieramente) en cuestión de días.
- **Desarrollar aplicaciones en el lenguaje que usted quiera.** Una vez que las aplicaciones han sido separadas de computadoras locales y sistemas operativos específicos éstas pueden también ser escritas en prácticamente cualquier lenguaje de programación. Ya que las aplicaciones web son esencialmente una colección de programas más que un simple programa, ellas podrían ser escritas en cualquier lenguaje de programación que esté por ahí. Mientras que para software escritorio usted está limitado a usar el mismo lenguaje que el sistema operativo subyacente este no es el caso cuando la aplicación de software es independiente del sistema operativo.

Desventajas

- Acceso limitado, la necesidad de conexión permanente y rápida a Internet hacen que el acceso a estas aplicaciones no esté al alcance de todos.
- La interactividad no se produce en tiempo real, en las aplicaciones web cada acción del usuario conlleva un tiempo de espera hasta que se obtiene la reacción del sistema.
- Elementos de interacción muy limitados. En comparación con el software de escritorio, las posibilidades de interacción con el usuario que ofrecen las aplicaciones web (mediante formularios principalmente) son muy escasas.
- Diferencias de presentación entre plataformas y navegadores. La falta de estándares ampliamente soportados dificulta el desarrollo de las aplicaciones.

De lo expuesto se puede decir que las aplicaciones web nos otorgan muchas ventajas, para el caso de las desventajas, la segunda y la tercera se solucionan escogiendo como marco de trabajo la plataforma JSF, que trabaja en base a eventos y nos proporciona una amplia gama de componentes reutilizables para el desarrollo de interfaces gráficas de usuario en ambientes web.

3.2.2 Servidor web

Un servidor web es un programa que atiende y responde a las diversas peticiones de los navegadores, proporcionándoles los recursos que solicitan mediante el protocolo HTTP o el protocolo HTTPS (la

versión segura, cifrada y autenticada de HTTP). Un servidor web básico tiene un esquema de funcionamiento muy sencillo, ejecutando de forma infinita el bucle siguiente:

1. Espera peticiones en el puerto TCP asignado (el estándar para HTTP es el 80).
2. Recibe una petición.
3. Busca el recurso en la cadena de petición.
4. Envía el recurso por la misma conexión por donde ha recibido la petición.
5. Vuelve al punto 2.

Un servidor web que siguiese el esquema anterior cumpliría los requisitos básicos de los servidores HTTP, aunque, eso sí, sólo podría servir ficheros estáticos.

A partir del esquema anterior se han diseñado y construido todos los programas servidores de HTTP que existen, variando sólo el tipo de peticiones (páginas estáticas, CGI, Servlets, etc.) que pueden atender, en función de que sean o no multi-proceso, multi-hilados, etc. En la sección 9.3 se detallan algunas de las características principales de los servidores web.

3.2.3 Aplicaciones web

Lo que el mercado demanda actualmente son mayormente aplicaciones web, que permitan a las empresas tradicionales llegar al cliente sin necesidad de que éste se desplace hasta la ubicación física

de la misma. Partiendo de este tipo de productos, además de la tecnología empleada para dar solución al problema, se han ido creando y perfeccionando distintas arquitecturas más o menos independientes de la tecnología aplicada. Dentro de esta familia de aplicaciones o herramientas, hay que destacar el papel que el lenguaje JAVA, junto con sus extensiones J2EE y el apoyo del mundo open-source están jugando.

Concepto. una aplicación web es una aplicación informática distribuida cuya interfaz de usuario es accesible desde un cliente web, normalmente un navegador web.

Características

– Comunicación mediante HTTP sobre TCP/IP.

– Procesamiento en servidor.

– Acceso a bases de datos.

– Arquitectura por capas.

– Distintos tipos de usuarios.

En la sección 9.4 se detallan los niveles de una aplicación web.

3.2.4 Servidor de aplicaciones

Un servidor de aplicaciones es un servidor web con capacidad de procesamiento, por lo que suele ser a la vez servidor web y servidor de lógica de negocio.

3.3 Definición de términos básicos.

3.3.1 Framework

Un framework representa una arquitectura de software que modela las relaciones generales de las entidades del dominio. Provee una estructura y una metodología de trabajo la cual extiende o utiliza las aplicaciones del dominio. También se conoce como marco de trabajo, en el presente trabajo se usará ambos términos indistintamente. Algunos ejemplos: Jakarta Struts, Microsoft .NET, Code Igniter.

3.3.2 JavaServer Faces (JSF)

Es un framework basado en Java para el desarrollo de aplicaciones Web que simplifica el desarrollo de interfaces de usuario para aplicaciones J2EE. En su versión estándar, JSF utiliza JavaServer Pages para su tecnología de visualización, pero JSF puede utilizar otras tecnologías, como XUL.

3.3.3 XUL

XUL (acrónimo de XML-based User-interface Language, lenguaje basado en XML para la interfaz de usuario) es la aplicación de XML a la descripción de la interfaz de usuario en el navegador Mozilla. XUL no es un estándar.

3.3.4 Servlet

Los servlets son objetos que corren dentro del contexto de un servidor web (Ejm.: Tomcat) y extienden su funcionalidad. También podrían correr dentro de un servidor de aplicaciones (Ejm.: OC4J Oracle) que además de

contenedor para servlet tendrá contenedor para objetos más avanzados como son los EJB's (Tomcat solo es un contenedor de servlets). La palabra servlet deriva de otra anterior, applet, que se refería a pequeños programas escritos en Java que se ejecutan en el contexto de un navegador web. Por contraposición, un servlet es un programa que se ejecuta en un servidor web

3.3.5 Swing

Swing es una Biblioteca gráfica para java que forma parte de las Java Foundation Classes (JFC). Incluye widgets para interfaz gráfica de usuario tales como cajas de texto, botones, desplegables y tablas.

CAPITULO IV:

4.- METODOLOGÍA DE LA INVESTIGACIÓN

4.1 Historia de las aplicaciones web

Inicialmente la web era simplemente una colección de páginas estáticas, documentos, etc., que podían consultarse o descargarse. El siguiente paso en su evolución fue la inclusión de un método para confeccionar páginas dinámicas que permitiesen que lo mostrado fuese dinámico (generado o calculado a partir de los datos de la petición). Dicho método fue conocido como CGI (common gateway interface) y definía un mecanismo mediante el cual podíamos pasar información entre el servidor HTTP y programas externos. Los CGI siguen siendo muy utilizados, puesto que la mayoría de los servidores web los soportan debido a su sencillez. Además, nos proporcionan total libertad a la hora de escoger el lenguaje de programación para desarrollarlos.

El esquema de funcionamiento de los CGI tenía un punto débil: cada vez que recibíamos una petición, el servidor web lanzaba un proceso que ejecutaba el programa CGI. Como, por otro lado, la mayoría de CGI estaban escritos en algún lenguaje interpretado (Perl, Python, etc.) o en algún lenguaje que

requería run-time environment (VisualBasic, Java, etc.), esto implicaba una gran carga para la máquina del servidor.

Además, si la web tenía muchos accesos al CGI, esto suponía problemas graves. Por ello se empiezan a desarrollar alternativas a los CGI para solucionar este grave problema de rendimiento. Las soluciones vienen principalmente por dos vías. Por un lado se diseñan sistemas de ejecución de módulos más integrados con el servidor, que evitan que éste tenga que ejecutar multitud de programas. La otra vía consiste en dotar al servidor de un intérprete de algún lenguaje de programación (RXML, PHP, VBScript, etc.) que nos permita incluir las páginas en el código de manera que el servidor sea quien lo ejecute, reduciendo así el tiempo de respuesta.

Ahora existen un gran número de arquitecturas y lenguajes de programación que nos permiten desarrollar aplicaciones web. Todas ellas siguen alguna de las dos vías ya mencionadas. De ellas, las más útiles y las que más se utilizan son aquellas que permiten mezclar los dos sistemas, es decir, un lenguaje de programación integrado que permita al servidor interpretar comandos que “incrustemos” en las páginas HTML y un sistema de ejecución de programas más enlazado con el servidor que no presente los problemas de rendimiento de los CGI.

La opción que quizás sea la más exitosa y potente de estas aproximaciones es la seguida por Sun Microsystems con su sistema Java, que está integrada por dos componentes; a saber, un lenguaje que permite incrustar código interpretable en las páginas HTML y que el servidor traduce a programas ejecutables, JSP (Java server pages) y un mecanismo de

programación estrechamente ligado al servidor, con un rendimiento muy superior a los CGI convencionales, llamado Java Servlet.

Otra de las tecnologías que más éxito ha obtenido y una de las que más se utiliza en Internet es el lenguaje de programación interpretado por el servidor PHP. Se trata de un lenguaje que permite incrustar HTML en los programas, con una sintaxis que proviene de C y Perl. Además, habida cuenta de su facilidad de aprendizaje, su sencillez y potencia, se está convirtiendo en una herramienta muy utilizada para algunos desarrollos.

Otros métodos de programación de aplicaciones web también tienen su mercado. Así sucede con mod_perl para Apache, RXML para Roxen, etc., pero muchos de ellos están vinculados a un servidor web concreto.

4.2 Framework Java Server Faces (JSF)

Actualmente existen dos técnicas populares para desarrollar aplicaciones web.

• El estilo del “desarrollo rápido”, usando un ambiente visual del desarrollo por ejemplo Microsoft ASP.NET.

• El estilo de “codificación hard-core”, escribiendo grandes números de líneas de código a fin de conseguir un alto desempeño de las aplicaciones por ejemplo J2EE (la edición empresarial de Java 2).

Los equipos de desarrollo enfrentan una decisión difícil. J2EE es una plataforma atractiva: altamente escalable, portable a múltiples plataformas y apoyada por muchos vendedores. Por otra parte, ASP.NET permite crear fácilmente atractivas interfaces de usuario sin una tediosa programación. Por

supuesto, los programadores desean ambos: un soporte de alto rendimiento y una programación fácil de interfaces de usuario. La promesa del framework Java Server Faces (JSF) es brindar un desarrollo rápido de interfaces de usuario del lado del servidor Java.

Si se está familiarizado con el desarrollo Java del lado del cliente, se puede pensar en JSF como “Swing para aplicaciones del lado del servidor.” Si se tiene experiencia con los JSP’s , Java Server Pages, encontrará que JSF proporciona muchas de las funcionalidades que los desarrolladores de JSP’s tienen que implementar a mano. Si se conoce ya un marco de desarrollo del lado del servidor tal como struts, encontrará que JSF utiliza una arquitectura similar.

4.6.1 ¿Qué es JSF?

La tecnología Java Server Faces (JSF) es un marco de trabajo de interfaces de usuario del lado de servidor para aplicaciones Web basadas en tecnología Java. Los dos componentes principales son: una librería de etiquetas para JSP y una API para manejo de eventos, validadores, etc.

Permite a los desarrolladores pensar en términos de componentes, eventos, beans manejadores y otras interacciones, en vez de hablar de peticiones, respuestas y marcas. JSF promete reutilización, separación de roles, facilidad de uso de las herramientas. JSF tiene una meta específica: hacer el desarrollo web más rápido y fácil.

4.6.2.- Tecnologías en que se basa

Se basa en las siguientes tecnologías: protocolo http, Servlets, JavaBeans, JSP.

- **Servlets**

Los Servlets son módulos que extienden los servidores orientados a petición-respuesta, como los servidores web compatibles con Java. Por ejemplo, un servlet podría ser responsable de tomar los datos de un formulario de entrada de pedidos en HTML y aplicarle la lógica de negocios utilizada para actualizar la base de datos de pedidos de la compañia.

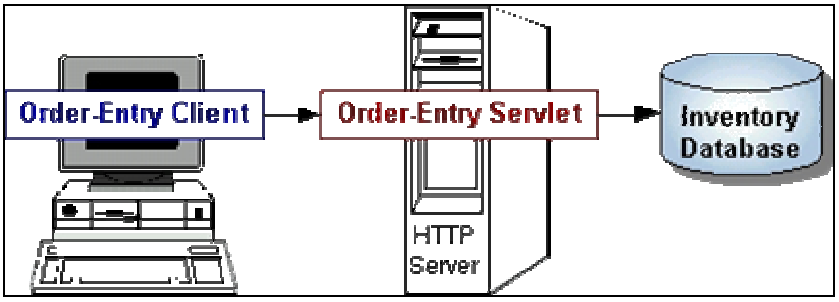

Fig. 1 Servlets

Los Servlets son para los servidores lo que los applets son para los navegadores. Sin embargo, al contrario que los applets, los servlets no tienen interface gráfico de usuario.

Los servelts pueden ser incluidos en muchos servidores diferentes porque el API Servlet, el que se utiliza para escribir Servlets, no asume nada sobre el entorno o protocolo del servidor. Los servlets se están utilizando ampliamente dentro de servidores HTTP; muchos servidores Web soportan el API Servlet.

Utilizar Servlets en lugar de Scripts CGI

Los Servlets son un reemplazo efectivo para los scripts CGI. Proporcionan una forma de generar documentos dinámicos que son fáciles de escribir y rápidos

en ejecutarse. Los Servlets también solucionan el problema de hacer la programación del lado del servidor con APIs específicos de la plataforma: están desarrollados con el API Java Servlet, una extensión estándard de Java.

Por eso se utilizan los servlets para manejar peticiones de cliente HTTP. Por ejemplo, tener un servlet procesando datos enviados sobre HTTP utilizando un formulario HTML, incluyendo datos del pedido o de la tarjeta de crédito. Un servlet como este podría ser parte de un sistema de procesamiento de pedidos, trabajando con bases de datos de productos e inventarios, y quizás un sistema de pago on-line.

Otros usos de los Servlets

Permitir la colaboración entre la gente. Un servlet puede manejar múltiples peticiones concurrentes, y puede sincronizarlas. Esto permite a los servlets soportar sistemas como conferencias on-line

Reenviar peticiones. Los Servlets pueden reenviar peticiones a otros servidores y servlets. Con esto los servlets pueden ser utilizados para balancear carga desde varios servidores que reflejan el mismo contenido, y para particionar un único servicio lógico en varios servidores, de acuerdo con los tipos de tareas o la organización compartida.

Inconvenientes

- Poco legible
- Mantenimiento costoso
- El diseñador gráfico debe saber Java
- A cada cambio: recompilar, empaquetar, desplegar.

- **Java Server Pages (JSP)**

JSP es la tecnología para generar páginas web de forma dinámica en el servidor, desarrollado por Sun Microsystems, basado en scripts que utilizan una variante del lenguaje java. La tecnología JSP, o de JavaServer Pages, es una tecnología Java que permite a los programadores generar dinámicamente HTML, XML o algún otro tipo de página web. Esta tecnología permite al código Java y a algunas acciones predefinidas ser embebidas en el contenido estático. En las JSP, se escribe el texto que va a ser devuelto en la salida (normalmente código HTML) incluyendo código java dentro de él para poder modificar o generar contenido dinámicamente. El código java se incluye dentro de las marcas de etiqueta <% y %>.

En una posterior especificación, se incluyeron taglib; esto es, la posibilidad de definir etiquetas nuevas que ejecuten código de clases java. La asociación de las etiquetas con las clases java se declaran en archivos de configuración en XML. La principal ventaja de JSP frente a otros lenguajes es que permite integrarse con clases Java (.class) lo que permite separar en niveles las aplicaciones web, almacenando en clases java las partes que consumen más recursos así como las que requieren más seguridad, y dejando la parte encargada de formatear el documento html en el archivo jsp. Además Java se caracteriza por ser un lenguaje que puede ejecutarse en cualquier sistema, lo que sumado a jsp le da mucha versatilidad.

Sin embargo JSP no se puede considerar un script al 100% ya que antes de ejecutarse el servidor web compila el script y genera un servlet, por lo tanto

se puede decir que aunque este proceso sea transparente para el programador no deja de ser una aplicación compilada. La ventaja de esto es algo más de rapidez y disponer del API de Java en su totalidad. Debido a esto la tecnología JSP, así como Java está teniendo mucho peso en el desarrollo web profesional (sobre todo en intranets).

Microsoft, la más directa competencia de Sun, ha visto en esta estrategia de Sun una amenaza lo que le ha llevado a que su plataforma .NET incluya su lenguaje de scripts ASP.NET que permite ser integrado con clases .NET (ya estén hechas en C++, VisualBasic o C#) del mismo modo que jsp se integra con clases Java.

JSF tiene las siguientes partes:

- Un conjunto de componentes de UI (Interfaz de usuario) prefabricados
- Un modelo de programación en base a eventos
- Un modelo de componentes que permite a los desarrolladores proveer componentes adicionales

JSF contiene todo el código necesario para el manejo de eventos y la organización de componentes. Los programadores de aplicaciones pueden ignorar estos detalles y concentrar su esfuerzo en la lógica de la aplicación. Para que la promesa de JSF sea completamente realizada, necesitamos ambientes de desarrollo integrado que generan aplicaciones JSF, mucho IDES están incorporando soporte JSF.

4.2.3 Análisis de una aplicación JSF

Las aplicaciones Web tienen dos partes: la capa de presentación y la lógica del negocio. La capa de presentación se refiere al aspecto de la aplicación. En el contexto de una aplicación basada en browser, el aspecto es determinado por las etiquetas HTML que especifican la disposición, fuentes, imágenes, y así sucesivamente. La lógica del negocio se implementa en el código Java que determina el comportamiento de la aplicación. Algunas tecnologías alternan HTML y código en un solo archivo. Ese acercamiento es popular puesto que es fácil producir aplicaciones simples en un solo archivo. Pero para aplicaciones serias, mezclar HTML y código trae problemas considerables. Los diseñadores web profesionales saben sobre diseño gráfico, pero por lo general confían en las herramientas que traducen su visión al HTML. Ciertamente no desearían ocuparse de código embebido. Por otra parte, los programadores no se ocupan del diseño gráfico. Así, para diseñar aplicaciones web profesionales, es importante separar la presentación de la lógica del negocio. Esto permite que los diseñadores y los programadores web se centren en sus capacidades principales. En el contexto de JSF, el código de la aplicación está contenido en beans, y el diseño en las páginas web.

Beans

Un bean Java es una clase que expone atributos y eventos a un ambiente por ejemplo JSF. Un atributo es un valor con nombre que puede ser leído y/o escrito. La manera más simple de definir un atributo es utilizar una convención de nombramiento estándar para los métodos de lectura y escritura, a saber, la familiar convención get/set. La primera letra del nombre del atributo

se cambia a mayúscula en los nombres del método. En aplicaciones JSF, se utiliza beans para todos los datos que necesiten ser accesibles desde una página. Las beans son los conductos entre la interfaz de usuario y el extremo final de la aplicación.

Páginas JSF

La implementación JSF define dos conjuntos de etiquetas. Las etiquetas core son independiente de la tecnología de representación. Por ejemplo, se necesita la etiqueta <f: view> tanto para páginas HTML como para las páginas que son renderizadas por un teléfono celular. Las etiquetas HTML generar marcas HTML específicas. Cuando se exhibe la página, los métodos get se invocan para obtener los valores actuales de los atributos. Cuando se envía la página, los métodos set se invocan para fijar los valores que el usuario incorporó.

Navegación

La navegación se especifica en el archivo faces-config.xml; a través de las reglas de navegación. Una regla de la navegación le dice a la implementación JSF qué página enviar de regreso al browser después de que se haya enviado un formulario. A continuación se muestra un ejemplo de una regla de navegación :

```
<navigation-rule>
<from-view-id>/index.jsp</from-view-id>
<navigation-case>
<from-outcome>login</from-outcome>
<to-view-id>/welcome.jsp</to-view-id>
</navigation-case>
<navigation-case>
<from-outcome>error</from-outcome>
<to-view-id>/error.jsp</to-view-id>
</navigation-case>
</navigation-rule>
```

Para el ejemplo anterior la navegación es simple. El usuario ingresa su usuario y password y si su ingreso de datos es exitoso será redireccionado a la página welcome.jsp, si su ingreso de datos es erróneo será redireccionado a la página error.jsp.

Además de las reglas de la navegación, el archivo de faces-config.xml contiene las definiciones de los beans. A continuación un ejemplo de cómo sería esta declaración :

```
<managed-bean>
<managed-bean-name>usuario</managed-bean-name>
<managed-bean-class>com.corejsf.UserBean</managed-bean-class>
<managed-bean-scope>session</managed-bean-scope>
</managed-bean>
```

En el ejemplo anterior se especifica el nombre del bean usuario, la clase en la que se implementan sus métodos y el alcance que puede tomar el valor request o session.

Configuración de Servlet

Cuando se despliega una aplicación JSF dentro de un servidor de aplicaciones, se necesita proveer un archivo de configuración llamado web.xml. Afortunadamente, se puede utilizar el mismo archivo web.xml para la mayoría de las aplicaciones JSF. El único aspecto a enfatizar de este archivo es el servlet mapping. Todas las páginas JSF son procesadas por un servlet especial que es una parte del código de implementación JSF. Para asegurar que el servlet correcto esté activado cuando se solicita una página JSF, las URLs JSF deben tener un formato especial. En nuestra configuración, tienen

una extensión .faces. Por ejemplo, no se puede señalar simplemente en el browser a http://localhost:8080/login/index.jsp. El URL tiene que ser http://localhost:8080/login/index.faces. El contenedor de servlets utiliza la regla servlet mapping para activar el servlet JSF, el que interpreta el sufijo faces y carga la página jsp equivalente.

4.2.4 .Ambientes visuales de desarrollo

Se puede construir las páginas JSF y los archivos de la configuración con un editor de textos. Sin embargo, se cuenta con que muchos programadores JSF utilicen ambientes visuales de desarrollo que llegan a estar una vez más disponibles. Un ambiente visual muestra una representación gráfica de los componentes y permite que un diseñador arrastre y deje caer componentes desde una paleta. La figura 2 muestra el IDE Sun Java Studio Creator (http://www.sun.com/software/products/jscreator). La paleta de componentes está en la esquina inferior-izquierda. Se pueden arrastrar los componentes sobre el centro de la ventana y modificarlos en el panel de propiedades en la esquina superior-derecha. El ambiente produce las etiquetas JSF correspondientes automáticamente.

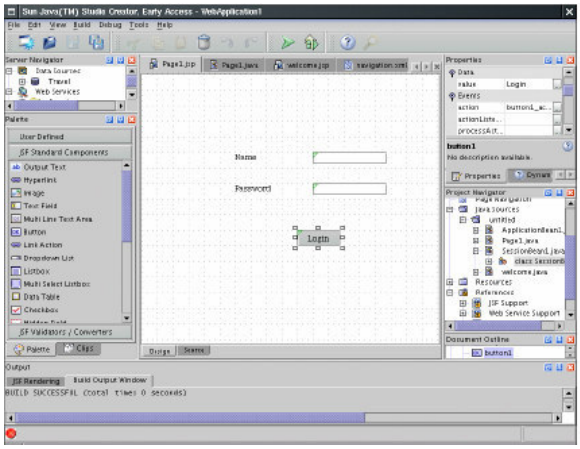

Fig. 2 Ambiente de desarrollo JSF visual

Por otra parte, los ambientes visuales brindan interfaces gráficas para especificar las reglas de la navegación y beans. Esos ambientes producen automáticamente el archivo faces-config.xml.

4.6.5 Servicios del framework JSF

La figura 3 da un visión de alto nivel de la arquitectura JSF. Como se puede ver, el framework JSF es responsable de la interacción con los dispositivos cliente, y proporciona las herramientas para vincular la presentación visual, la lógica de la aplicación, y la lógica del negocio de una aplicación web. Sin embargo, el alcance de JSF se restringe a la capa de presentación. La persistencia de la base de datos, los servicios web, y otras conexiones están fuera del alcance de JSF.

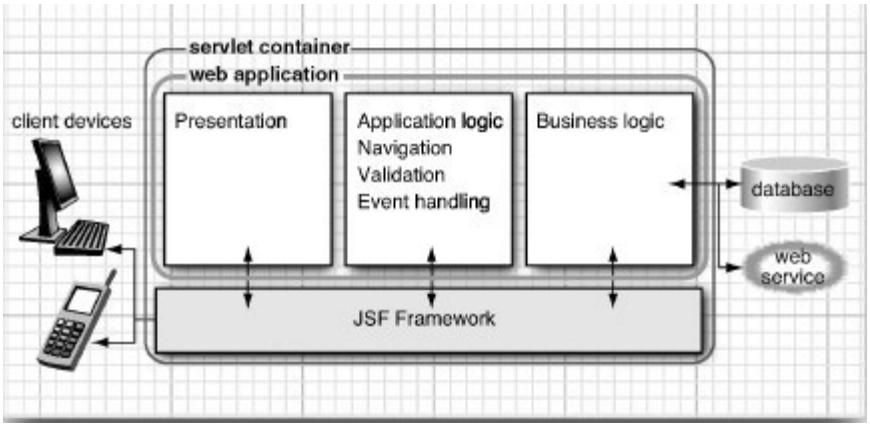

Fig. 3 Vista de alto nivel del framework JSF

Aquí están los servicios más importantes que el framework JSF proporciona.

Arquitectura Modelo-Vista-Controlador

Todas las aplicaciones de software permiten a los usuarios manipular cierta data, tales como carros de compras, itinerarios de viajes, o cualquier dato requerido en un particular contexto. Estos datos se llaman el modelo. Justo

como un artista crea una pintura de un modelo en un estudio, un desarrollador del software produce vistas del modelo de datos. En una aplicación web, el HTML (o una tecnología de representación similar) se utiliza para pintar estas vistas. JSF conecta la vista y el modelo. Un componente de la vista puede ser relacionado con un atributo de un bean de un objeto del modelo, por ejemplo <h: inputText value= " # {user.name} "/>, donde se asocia el atributo name del bean user con una caja de texto.

Por otra parte, JSF funciona como el controlador que reacciona al usuario procesando acciones y eventos de cambio de valor, encaminándolos al código que actualiza el modelo o la vista. Por ejemplo, se puede desear invocar un método para comprobar si se permite a un usuario abrir una sesión. Utilizar la siguiente etiqueta JSF:

```
<h:commandButton value="Login" action="#{user.check}"/>
```

Cuando se presiona el botón y el formulario se envía al servidor, la implementación JSF invoca el método check del bean user. Ese método puede tomar acciones arbitrarias para actualizar el modelo, y retorna la identificación de la navegación de la siguiente página que se mostrará. Así, JSF implementa la clásica arquitectura Modelo-Vista-Controlador.

Conversión de datos

Los usuarios incorporan datos en los formularios web como texto. Los objetos del negocio desean datos como números, fechas, u otros tipos de datos. JSF hace fácil especificar y modificar reglas de conversión.

Validación y gestión de errores

JSF hace fácil unir reglas de validación para los campos tales como "este campo es requerido" o "este campo debe ser un número." Por supuesto, cuando los usuarios incorporan datos inválidos, se necesita exhibir mensajes de error apropiados. JSF minimiza mucho del tedio de esta tarea de programación.

Internacionalización

JSF maneja ediciones de internacionalización tales como codificaciones de caracteres y la selección de resource bundles.

Componentes comunes

Los desarrolladores de componentes pueden desarrollar sofisticados componentes que diseñadores de página simplemente utilicen en sus páginas.

Renderizador Alternativos

Por defecto, JSF genera páginas HTML. Pero es fácil extender el framework JSF para producir etiquetas para otro lenguaje de descripción de páginas tal como WML o XUL.

Herramientas de soporte

JSF se optimiza para el uso con herramientas automatizadas. Como estas herramientas maduraran en los años que vienen, se cree que JSF será el framework para el desarrollo de interfaces web con Java.

4.6.6 Como trabaja el framework JSF

Cuando el browser se conecta con por ejemplo http://localhost:8080/login/index.faces, el servlet de JSF inicializa el código JSF y lee la página index.jsp. La página contiene etiquetas tales como f: form y h: inputText. Cada etiqueta tiene asociada una clase tag handler. Cuando se lee la página, los tag handler se ejecutan. Los tag handler JSF colaboran con cada uno para construir un árbol de componentes.

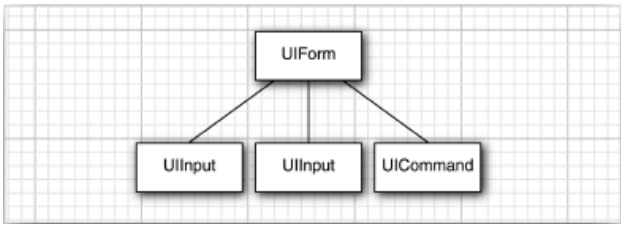

Fig. 4 Árbol de componentes

Un árbol de componentes es una estructura de datos que contiene objetos Java para todos los elementos de la interfaz de usuario en la página JSF. Por ejemplo, los dos objetos UIInput corresponden a dos cajas de texto, y el objeto UICommand a un botón del archivo JSF.

Representación de las páginas

Después, la página HTML es renderizada . Todo el texto que no es una etiqueta JSF se pasa simplemente por alto y las etiquetas JSF son convertidas a HTML. Cada una de estas etiquetas da lugar a un componente asociado. Cada componente tiene un renderizador que produce salida HTML, reflejando el estado del componente. Por ejemplo, el renderizador para el componente que corresponde a la etiqueta h:inputText produce la siguiente salida:

```
<input type="text" name="unique ID" value="current value"/>
```

Este proceso se llama codificación. El renderizador del objeto UIInput pide al framework buscar el ID único y el valor actual de la expresiones del tipo bean.atributo Por defecto, las secuencias de ID's (tales como _id0: _id1) son asignados por el framework. La página codificada se envía al browser, y el browser lo muestra en manera usual.

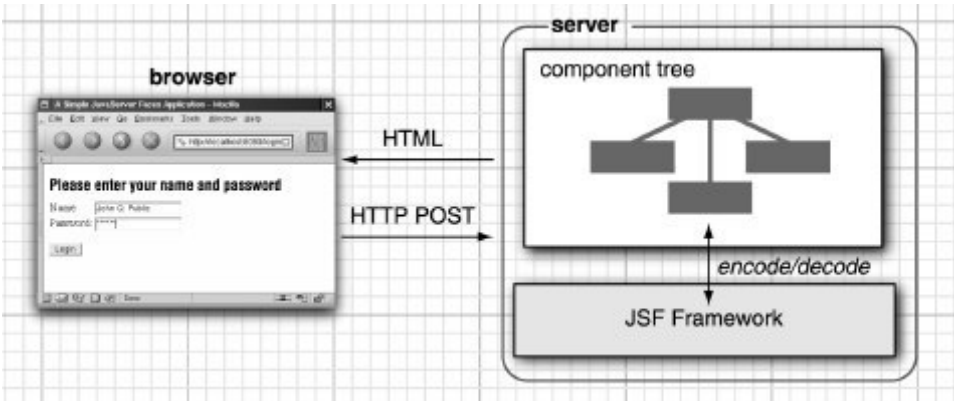

Fig. 5 Codificando y decodificando páginas JSF

Decodificando peticiones

Después de que la página se muestra en el browser, el usuario completa los campos del formulario y pulsa el botón "Sign in". El browser envía los datos del formulario de nuevo al servidor web, formateado como una "petición POST." Este es un formato especial, definido como parte del protocolo HTTP. La petición POST contiene el URL del formulario (/login/index.faces), así como los datos del formulario. La data del formulario es una cadena de pares ID/valor, por _id0:_id1=me&_id0:_id2=secret&_id0:_id3=Login&_id0=_id0. Como parte del normal procesamiento del servlet, la data del formulario se ponen en una tabla hash que todos los componentes pueden tener acceso. Después, el framework JSF da a cada componente una oportunidad de examinar tabla hash, un proceso llamado decodificación. Cada componente decide cómo interpretar los datos del formulario.

El componente UIInput actualiza los atributos del bean referenciados en el atributo value : invocan los métodos setter con los valores que el usuario ingresó. El componente UICommand comprueba si el botón fuera pulsado. Si es así él enciende un evento para lanzar la acción login referida en el atributo action. Ese evento indica la siguiente página a mostrar, para el ejemplo welcome.jsp. Ahora el ciclo se repite, con la construcción del árbol de componentes para la página.

4.6.7 El ciclo de vida JSF

La especificación JSF define seis fases distintas, según como se muestra en la figura 6. El flujo normal del control se muestra con las líneas sólidas; los flujos alternativos se muestran con las líneas discontinuas.

La fase *Restore View* recupera el árbol componente para la página solicitada si fue exhibida previamente o construye un nuevo componente árbol si se exhibe por primera vez. Si la página fue exhibida previamente, todos los componentes se fijan a su estado anterior. Esto significa que JSF conserva automáticamente la información del formulario. Por ejemplo, cuando un usuario envía data errónea que se rechazan durante la decodificación, los viejos campos de texto se vuelven a mostrar de modo que el usuario pueda corregirlas.

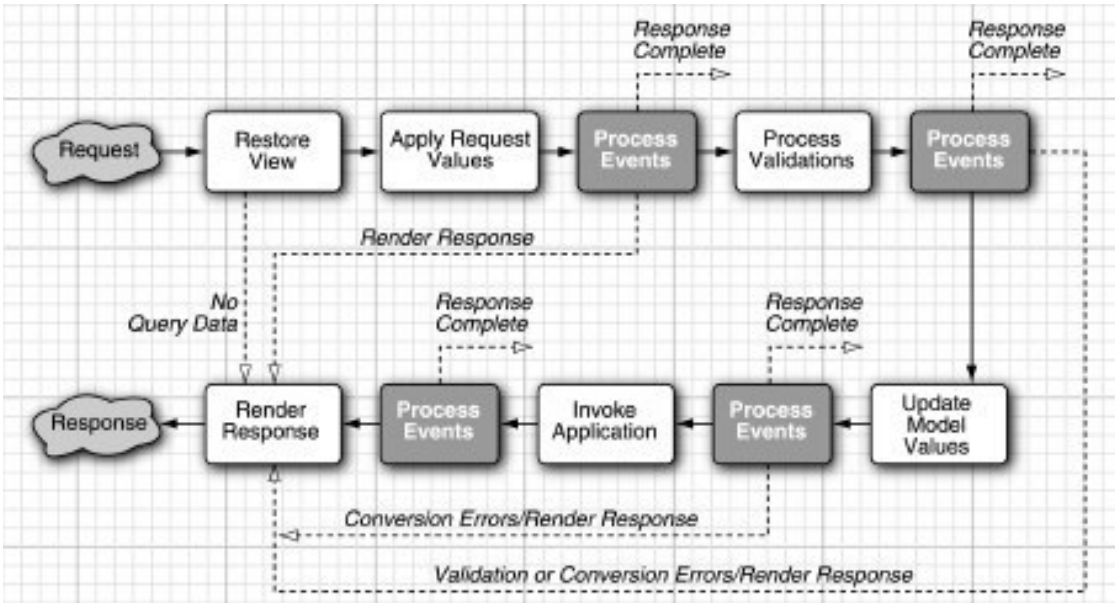

Fig. 6 Ciclo de vida de JSF

Si la petición no tiene ningún dato de la consulta, la implementación JSF salta a continuación a la fase *Render Response*. Esto sucede cuando una página se exhibe por primera vez. Si no, la fase próxima es la fase *Apply Request Values*. En esta fase, la implementación JSF itera sobre los objetos componente en el componente árbol. Cada objeto componente chequea que valores solicitados pertenecen a éste y los almacena. En la fase *Process Validations*, los valores enviados primero se convierten a "valores locales," que pueden ser objetos de cualquier tipo. Cuando diseñas una página JSF, puedes unir validadores que comprueban la validez de los valores locales enviados. Si la validación pasa, el ciclo de vida JSF procede normalmente. Sin embargo, cuando ocurren errores de conversión o validación, la implementación JSF invoca la fase Render Response directamente, volviendo a mostrar la página actual de modo que el usuario tenga otra opción de proporcionar entradas correctas

Después que los convertidores y los validadores han hecho su trabajo, se asume que es seguro actualizar el modelo de datos. Durante la fase *Update Model*, los valores locales se utilizan para actualizar los beans que se relacionan a los componentes.

En la fase *Invoke Application*, el método action de un botón o un link que causó el envío del formulario es ejecutado. Ese método puede realizar algún procesamiento de la aplicación arbitrariamente. Este retorna una cadena de salida que se pasa al tratante de la navegación. El tratante de la navegación muestra la página siguiente. Finalmente, la fase *Render Response* codifica la respuesta y la envía al browser. Cuando un usuario acepta o envía un formulario, pulsa un link, o de otra manera genera una nueva petición, el ciclo comienza de nuevo.

CAPITULO V

5.- SOLUCION IM PLEMENTADA

5.1 Descripción de la institución.

El Consejo Nacional de Ciencia, Tecnología e Innovación Tecnológica (CONCYTEC), es una institución gubernamental, tiene como uno de sus objetivos motivar e incentivar la vocación por las ciencias, promover el desarrollo de las habilidades y capacidad creativa de los recursos humanos, así como la difusión de los conocimientos y avances científicos y tecnológicos a nivel nacional, para el cumplimiento de dicho objetivo otorga subvenciones periódicamente bajo diferentes categorías: Becas, Tesis, Premios, Eventos, Publicaciones y Proyectos.

5.1.1 Visión

El CONCYTEC cuenta con el reconocimiento y apoyo de la sociedad peruana por su contribución al desarrollo nacional; lidera la acción promotora del Sistema Nacional de Ciencia, Tecnología e Innovación Tecnológica, en todo el país, logrando el concurso plural y concertado de la academia, el Estado, la empresa y la colectividad en el esfuerzo sistemático por el desarrollo humano

basado en el conocimiento, que favorece el bienestar del país y su participación en el avance científico mundial.

5.1.2 Misión

Proveer los instrumentos político-normativos y técnicos para generar las condiciones propicias al desarrollo de la creatividad y la capacidad innovadora del país, mediante el fortalecimiento de la institucionalidad y del Sistema Nacional de Ciencia, Tecnología e Innovación Tecnológica (SINACYT), articulando la acción del Estado-Academia-Empresa y Sociedad, para la movilización de iniciativas, talentos y capacidades en el esfuerzo por generar, captar, adaptar, transferir y difundir el conocimiento científico y tecnológico que el país requiere para su inserción en la sociedad del conocimiento, como una nación emergente, próspera, equitativa y solidaria, genuinamente comprometida con el desarrollo humano sustentable.

5.1.3 Funciones

De acuerdo a la 3ra. disposición transitoria de la Ley 28303:

Son funciones del CONCYTEC:

a) Normar, dirigir, orientar, coordinar y articular el SINACYT, así como el proceso de planeamiento, programación, seguimiento y evaluación de las actividades de CTeI (Ciencia, Tecnología e Innovación).

b) Formular la política y planes nacionales de desarrollo científico y tecnológico, articulando las propuestas sectoriales, regionales e

institucionales de CTeI, con los planes de desarrollo socioeconómico, ambientales y culturales del país.

c) Promover la descentralización y adaptación de las actividades de CTeI en el ámbito regional y local.

d) Coordinar con los sectores y entidades del Estado y sector privado, sus planes estratégicos sectoriales y planes operativos institucionales, a fin de articularlos con el Plan Nacional de CTeI y de lograr la interconexión progresiva de sus sistemas de información en una red nacional de información científica e interconexión telemática.

e) Promover y desarrollar mecanismos de protección de los derechos de propiedad intelectual, propiedad industrial y sus derechos conexos en coordinación con los organismos competentes.

f) Promover y desarrollar mecanismos de protección del conocimiento tradicional y fomentar el rescate, utilización y difusión de las tecnologías tradicionales en coordinación con los organismos competentes.

g) Brindar asesoría a las instancias del Gobierno y a los poderes del Estado en materia de CTeI

h) Promover la articulación de la investigación científica tecnológica, y la producción del conocimiento con los diversos agentes económicos y

sociales, para el mejoramiento de la calidad de vida y el impulso de la productividad y competitividad del país.

i) Implementar mecanismos de coordinación, intercambio y concertación entre las instituciones integrantes del SINACYT, así como con el empresariado, universidades, embajadas y otras entidades del país y del exterior.

j) Aprobar los programas nacionales y compatibilizar los programas regionales especiales de CTeI.

k) Desarrollar y ejecutar programas especiales de CTeI orientados a la formación, perfeccionamiento, retención y colaboración de científicos y tecnólogos, así como para el apoyo de la investigación universitaria y para la promoción de proyectos de innovación, transferencia, difusión, intercambio y divulgación de la CTeI.

l) Coordinar con las entidades competentes la recopilación, sistematización y control de calidad de la información e indicadores de CTeI, los procedimientos de normalización, calificación y registro de entidades de CTeI, concursos de méritos, premios, licitaciones, contratos y convenios de CTeI.

m) Diseñar y proponer a las instancias correspondientes las normas y estrategias para el cumplimiento de los objetivos de la presente ley, así

como la reglamentación y directivas para la implementación del esquema promocional y régimen de incentivos.

n) Elaborar los informes periódicos sobre el estado de situación general de la CTeI y sobre el avance de la ejecución presupuestal respectiva, así como los informes de evaluación sobre el desempeño de las entidades integrantes del Sistema.

o) Proponer la asignación de recursos disponibles y el régimen de incentivos en CTeI, de acuerdo a ley

p) Diseñar las políticas sobre transferencia de tecnología, así como los mecanismos de cooperación con otros países y organismos internacionales en materia de CTeI.

q) Calificar a las instituciones e investigadores que conforman el SINACYT, de acuerdo a lo que establezca el reglamento de la presente ley.

r) Emitir opinión sobre proyectos normativos o institucionales vinculados con la CTeI.

s) Promover el estudio del conocimiento y las tecnologías tradicionales.

t) Promover el establecimiento y desarrollo de una red nacional de información científica e interconexión telemática, para un manejo ágil,

oportuno y eficiente de la estadística científico-tecnológica que permita la obtención de la información necesaria para el planeamiento, operación y promoción de CTeI.

u) Otras establecidas por ley.

5.2 Análisis del negocio

5.2.1 Procesos del negocio

Como se ha expuesto anteriormente, las subvenciones otorgadas por Concytec son de diferentes tipos, a manera de explicar el proceso de una subvención se tomará como ejemplo las subvenciones de becas para estudios de postgrado en universidades peruanas.

a) La Convocatoria

Dependencias involucradas: DGAI (Dirección General de Apoyo al Investigador).

Las convocatorias de Becas para Estudios de Postgrado en Universidades Peruanas se aperturan una vez por año entre los meses de marzo, Abril y Mayo generalmente.

Cada convocatoria se rige por su reglamento, el cual es un documento que indica los procedimientos a seguir para una postulación a la convocatoria, los requisitos que debe presentar la persona que solicita la subvención, la forma como se evaluará las subvenciones y los formatos que debe ser completados por el solicitante. Este reglamento varía año tras año, así como los requisitos exigidos, no existe un formato único y los datos de la persona también son solicitados en diferente detalle convocatoria tras convocatoria.

Procedimiento para presentarse a la convocatoria

Los interesados en participar en el concurso “Becas para Estudios de Postgrado en Universidades Peruanas”, en convocatorias proceden de una de las siguientes formas:

** Enviar su expediente a las oficinas descentralizadas del CONCYEC ubicados en universidades del interior del país.

** Enviar su expediente directamente en mesa de partes del CONCYTEC.

b) La evaluación

Dependencias involucradas: DGAI (Dirección General de Apoyo al Investigador), comité AdHoc, comité directivo.

Los procesos de evaluación importantes comienzan con la recepción y revisión de los expedientes, se verifica que todos los solicitantes estén aptos para la evaluación (verificación de requisitos exigidos), luego se procede a clasificarlos y se obtienen estadísticas iniciales.

Según clasificación por áreas, se procede a distribuir expedientes a evaluadores pares, los cuales según cartilla de evaluación, procederán a evaluar. Con los resultados de las evaluaciones se procede a consolidar las calificaciones y obtener listados por puntajes obtenidos y otros mas. Estos listados son evaluados por el comité AdHoc y en ultima instancia es aprobado la lista de ganadores por el comité directivo.

c) Contratación

Dependencias involucradas: DGAI (Dirección General de Apoyo al Investigador), Asesoría Legal, Administración, Tesorería.

El departamento de Asesoría Legal proporciona una serie de números correlativos válidos para la elaboración de contratos a DGAI, dependencia que es la encargada de la elaboración de los contratos. Luego que los contratos son elaborados deben ser validados por Asesoría legal, si el contrato no es válido DGAI lo corrige, si es válido se procede a la firma del contrato para su posterior entrega al subvencionado. Luego se genera un documento llamado F1, es una solicitud de gasto donde se especifica el monto a otorgar, en que fecha y los datos del subvencionado.

En Administración, se procede a elaborar la orden de subvención respectiva, para la generación del comprobante de pago y la emisión del cheque. Tesorería con la orden de subvención aprobada genera los comprobantes de pago y los cheques, los subvencionados deben recoger el cheque en Tesorería.

d) Seguimiento

Dependencias involucradas: DGAI (Dirección General de Apoyo al Investigador), Administración.

Los subvencionados están obligados a presentar rendiciones del dinero recibido. Las fechas de entrega de estas rendiciones están estipuladas en su contrato. El siguiente desembolso de dinero está supeditado a que el subvencionado haya rendido adecuadamente.

5.2.2 Flujo de datos

A continuación se presenta como es el flujo de datos en el proceso de entrega de subvenciones, particularmente de las subvenciones en becas correspondiente al año 2003.

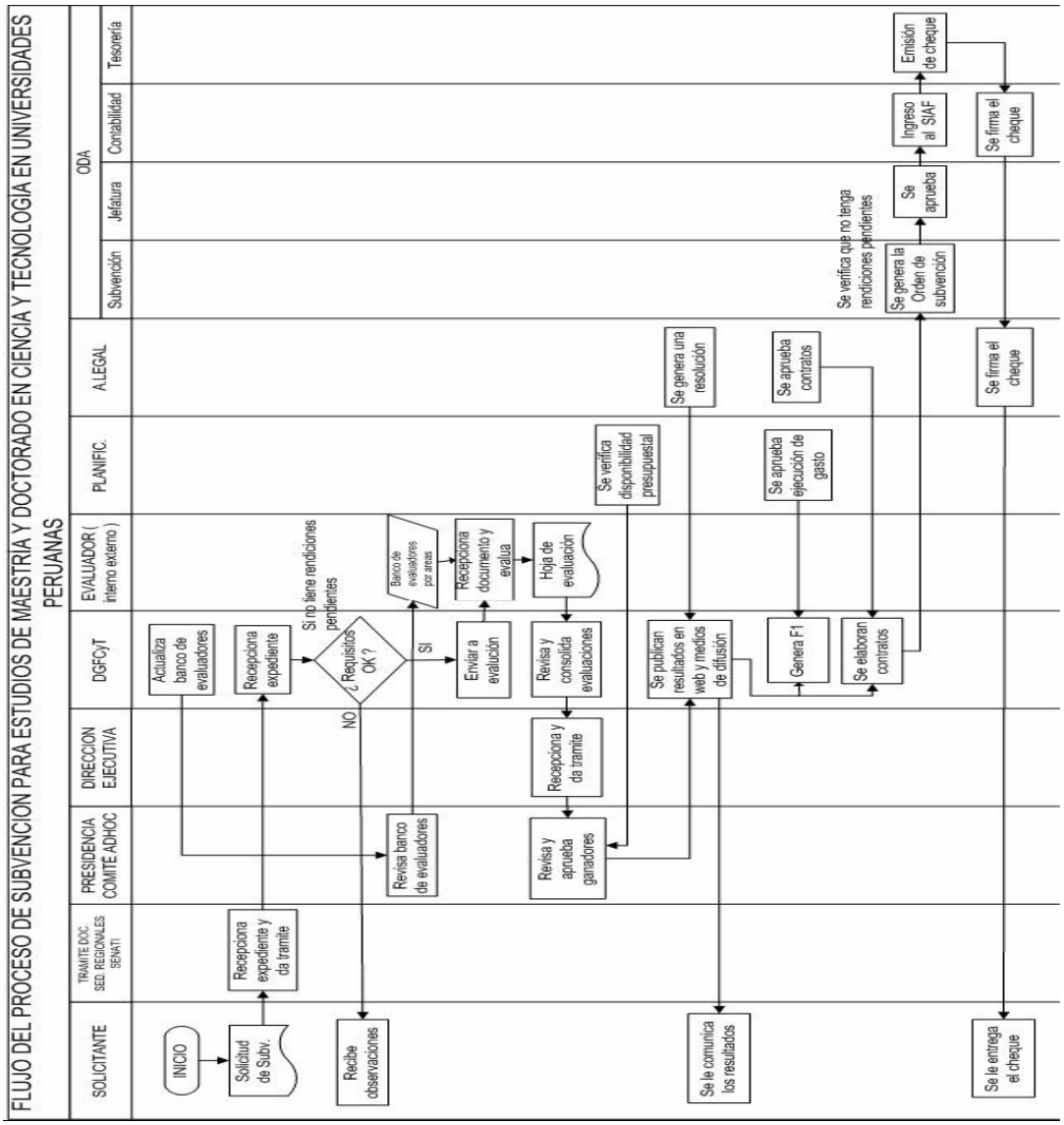

Fig. 7 Flujo de datos – Becas 2003

Leyenda:

DGFCYT: Dirección General de Formación en Ciencia y Tecnología.

SIAF: Sistema Integral de Administración Financiera

5.2.3 Flujo de documentos

Un ciclo comienza cuando inicia una convocatoria para una subvención ó beca, luego existe un periodo de tiempo para la recepción ó inscripción de postulantes (y una revisión inicial), al término de la convocatoria, se procede a consolidar los expedientes recibidos y depurar los que no cumplan los requisitos exigidos.

Posteriormente se clasifica los expedientes y se obtienen estadísticas iniciales, esta clasificación (por área temática) también servirá para asignar evaluadores (pares) y se procede a convocar a los evaluadores para el proceso de evaluación. Como resultado del proceso de evaluación se obtienen listados y se procede a su aprobación por parte del comité directivo y su publicación. Aqui se inicia el tratamiento administrativo para la ejecución del gasto y posterior seguimiento, para esto se coordina y aprueba en planificación, dirección ejecutiva y administración.

En el siguiente cuadro se podrá apreciar en forma grafica esta secuencia.

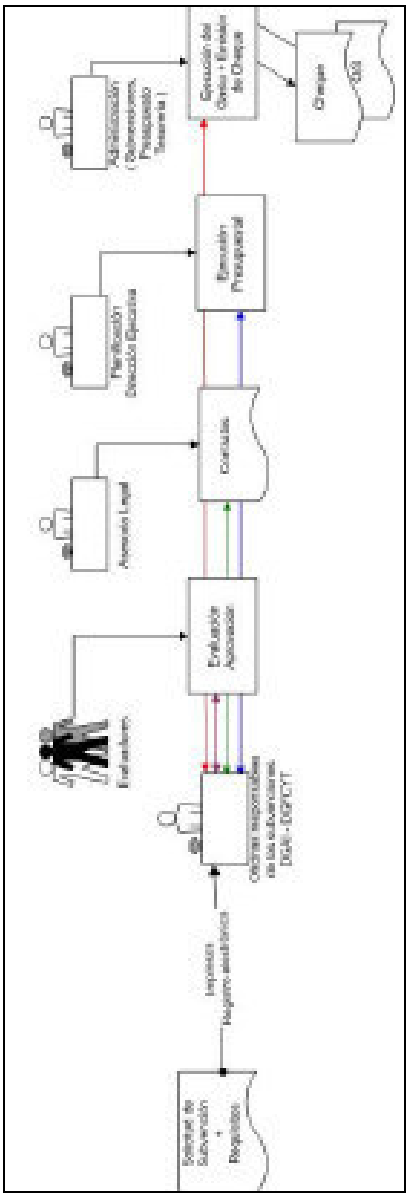

Fig. 8 Flujo de información subvenciones

Leyenda:

DGFCYT: Dirección General de Formación en Ciencia y Tecnología.

DGAI: Dirección General de Apoyo a la Investigación.

O/S: orden de subvención.

5.3 Sistema actual: “Sistema de Subvenciones”

5.3.1 Antecedentes

Se hicieron intentos aislados de construir el sistema de subvenciones desde 1999. A inicios del 2002 se iniciaron las conversaciones para la construcción del sistema de subvenciones. A mediados del 2002, debido a la priorización de otras actividades y falta de recursos para el desarrollo de software se abandonó la idea del desarrollo de un sistema integral de subvenciones.

A finales del 2002 se retoma el tema con apoyo de nuevo personal. Durante los ultimos meses del 2002 y los primeros 4 meses del 2003 se construyen los primeros módulos del sistema de subvenciones que actualmente se encuentran en uso (Proyectos, Eventos,Publicaciones). Se prueba el sistema de proyectos A mediados del 2003 se reduce la cantidad de personal asignado al proyecto y se alargan los tiempos de desarrollo, pese a ello se termina los módulos de subvenciones para administración.

A finales del 2003 ya no se cuenta con equipo de desarrollo para el sistema de subvenciones. A pesar de ello se registra en el sistema las subvenciones de años anteriores (4 ultimos años) con el apoyo de dos digitadores contratados por 1 mes. En el 2003 también se construyen los módulos cliente para Tesis y Becas en su fase inicial. A inicios del 2004 se inicia la construcción del módulo de premios (nuevo tipo de subvención).

El sistema actual se usa en Administración pero en DGAI (Dirección General de Apoyo al Investigador) de 14 usuarios sólo 1 lo usa debido a su poca facilidad para el desarrollo de tareas.

5.3.2 Características

El sistema actual es una aplicación de escritorio, bajo la siguiente plataforma de desarrollo:

- Lenguaje de Programación: Visual Basic 6.0
- Base de Datos: MS Sql Server 2000

5.3.3 Módulos

La plataforma está constituida por los siguientes módulos:

- Módulo Becas
- Módulo Proyectos
- Módulo Tesis
- Módulo Eventos
- Módulo Premios
- Módulo Publicaciones
- Módulo Orden de subvención

5.3.4 Interfaces de usuario

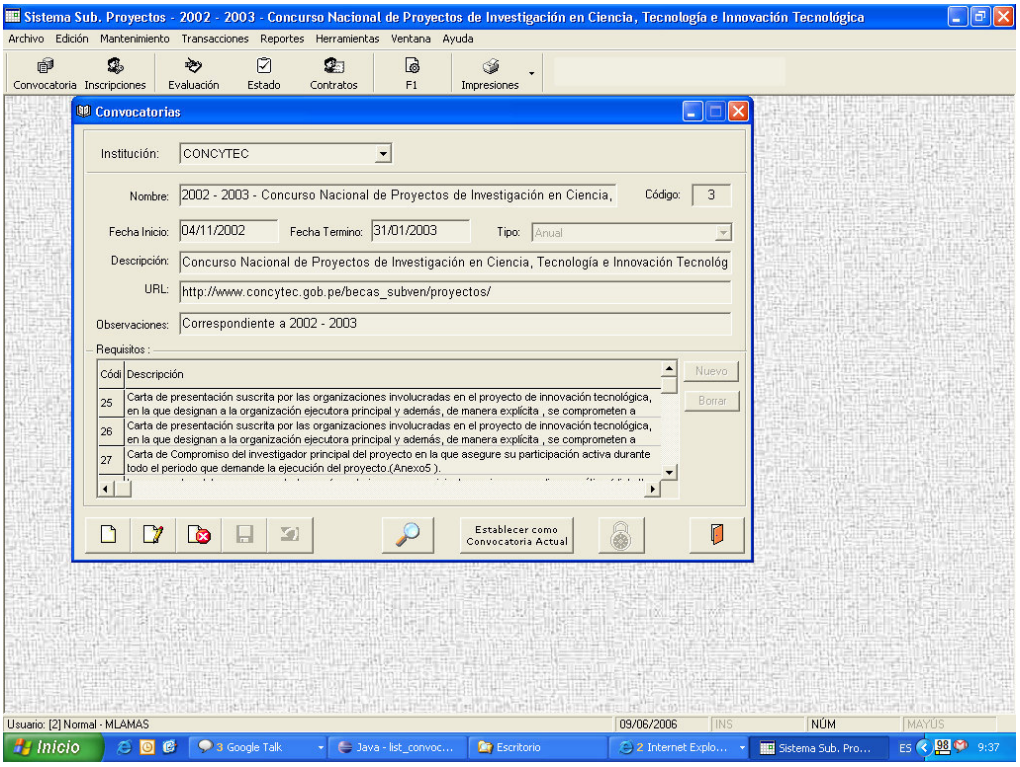

Fig. 9 Sistema actual –Becas:Convocatoria

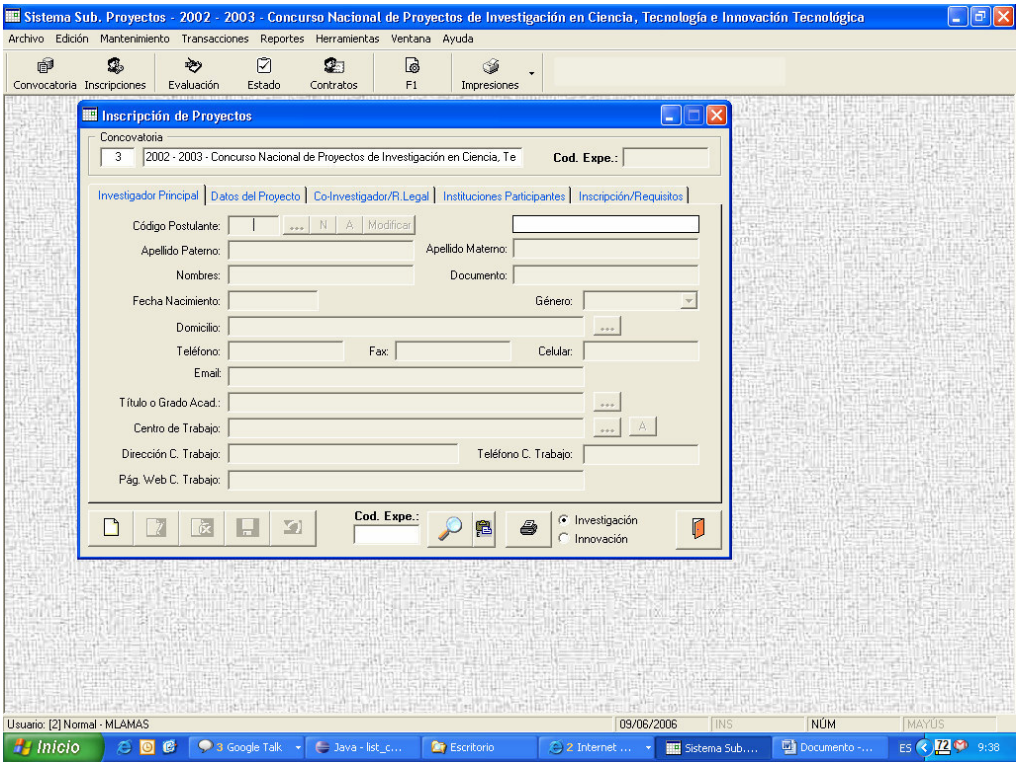

Fig. 10 Sistema actual –Becas: inscripciones

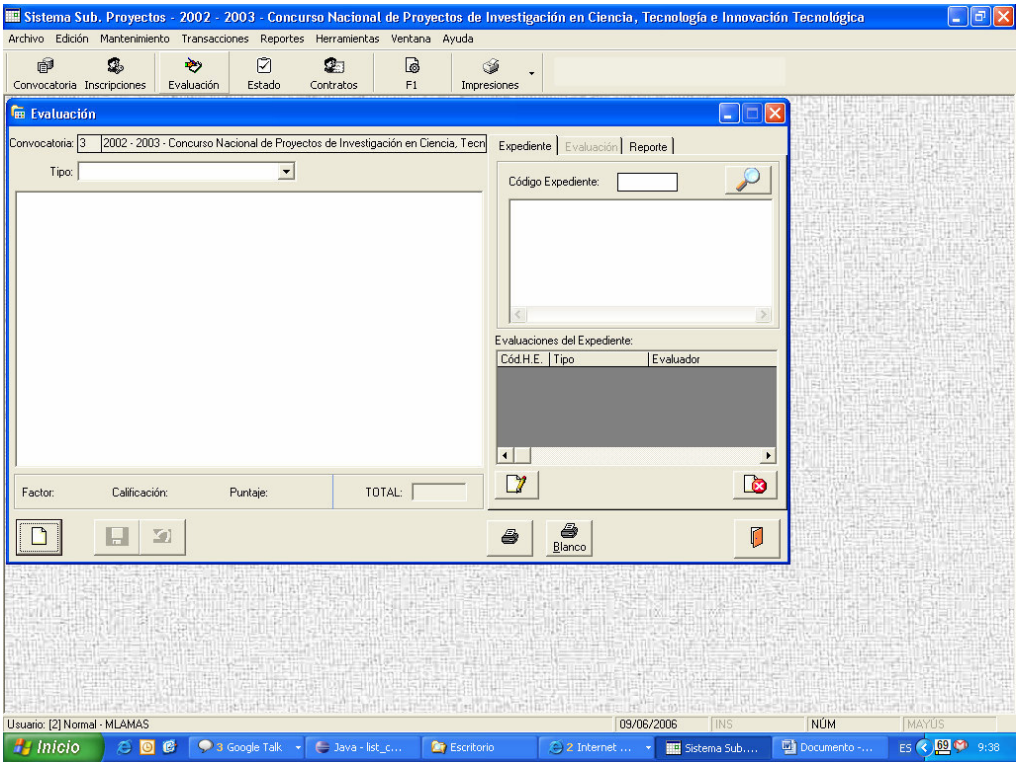

Fig. 11 Sistema actual –Becas: Evaluación

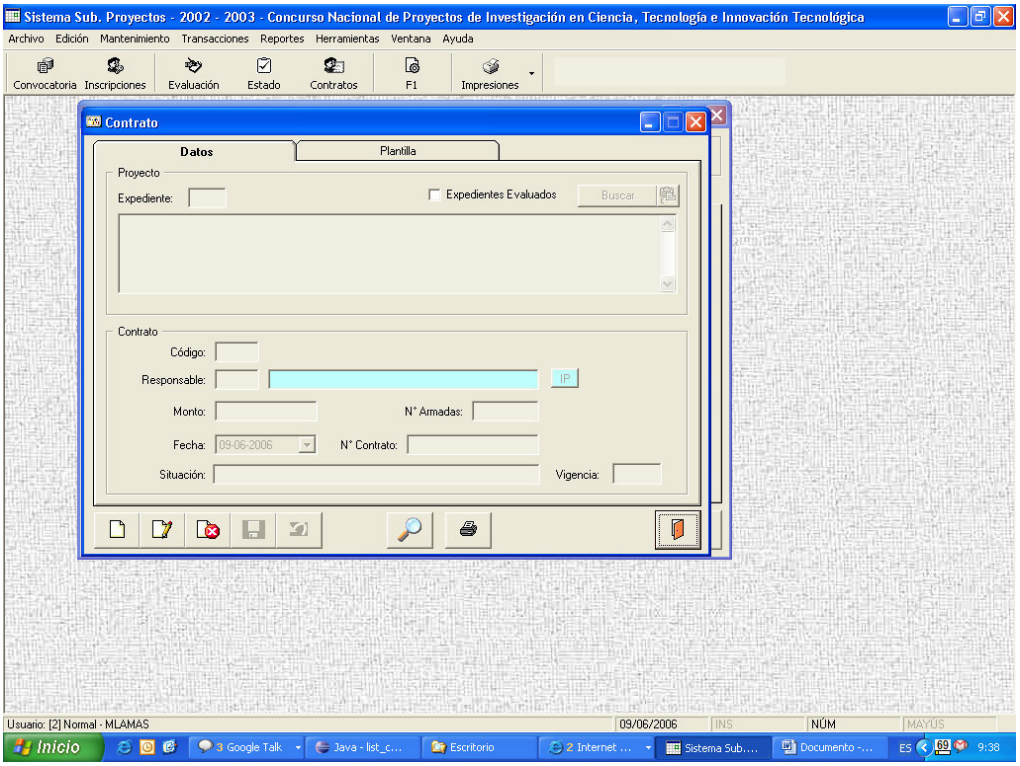

Fig. 12 Sistema actual –Becas: contratos

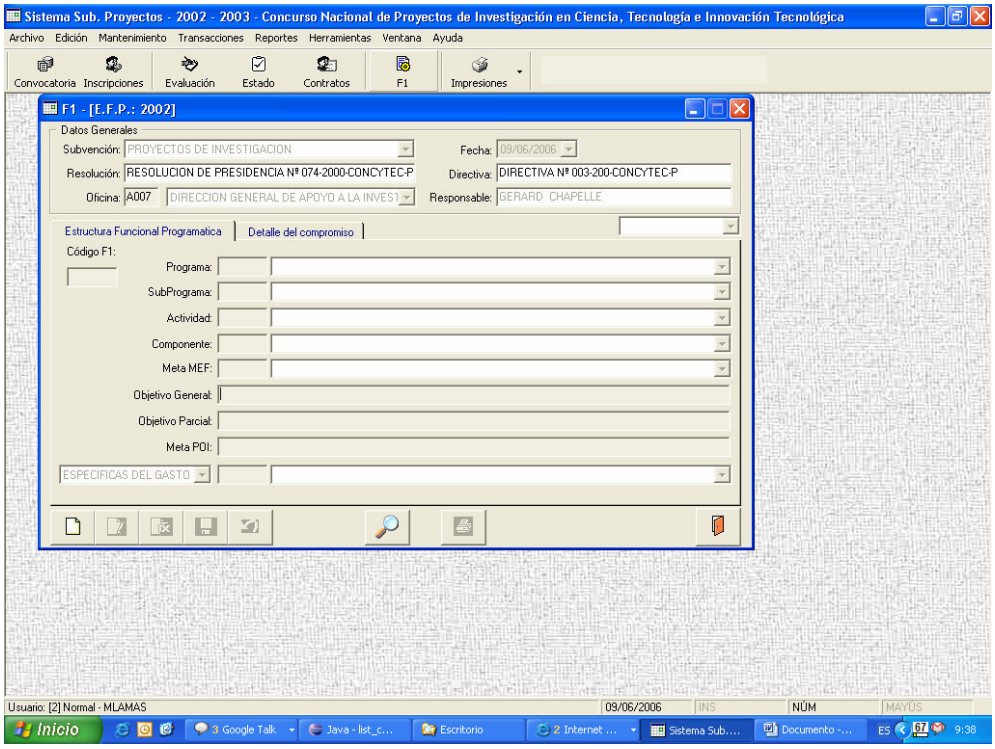

Fig. 13 Sistema actual –Becas: generación F1

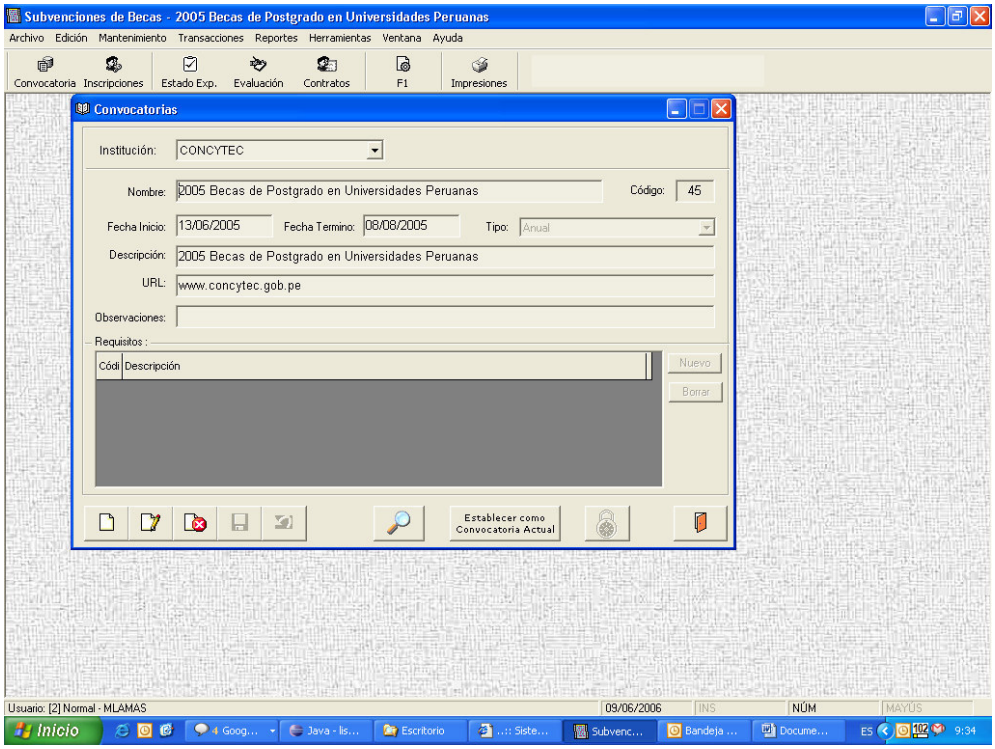

Fig. 14 Sistema actual –Proyectos: convocatoria

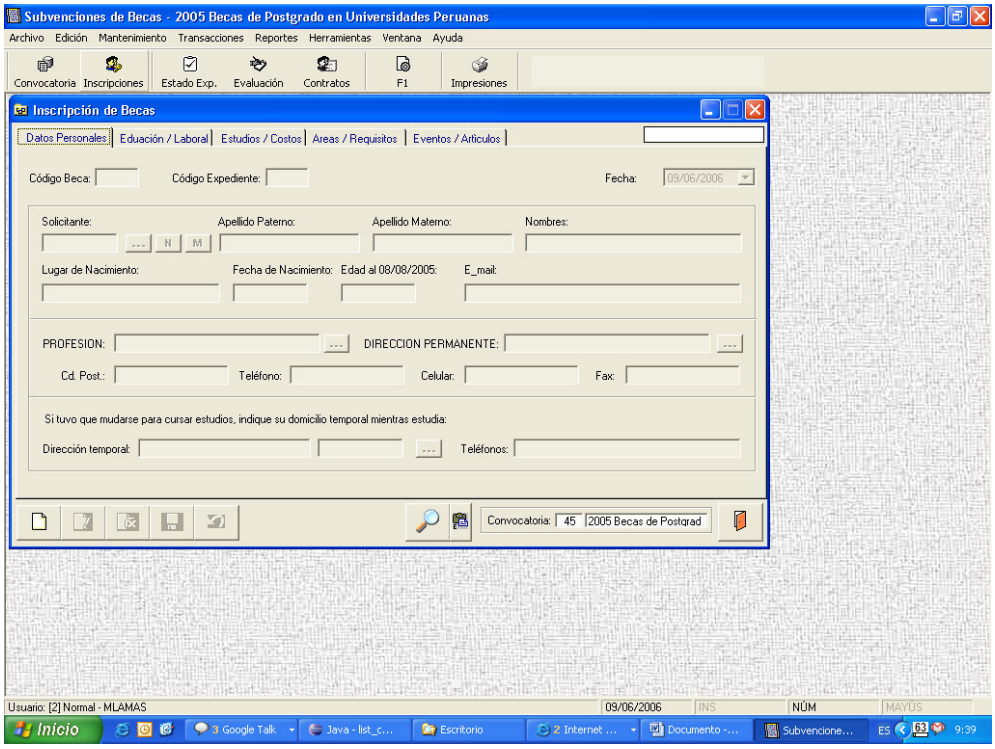

Fig. 15 Sistema actual –Proyectos: inscripcione

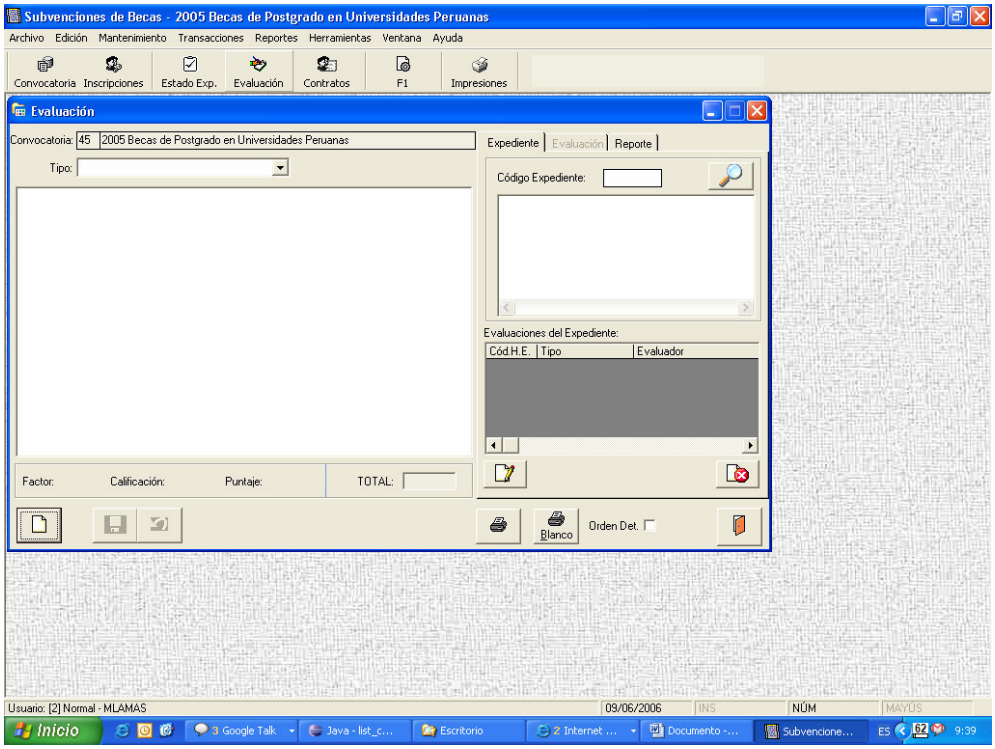

Fig. 16 Sistema actual –Proyectos: evaluación

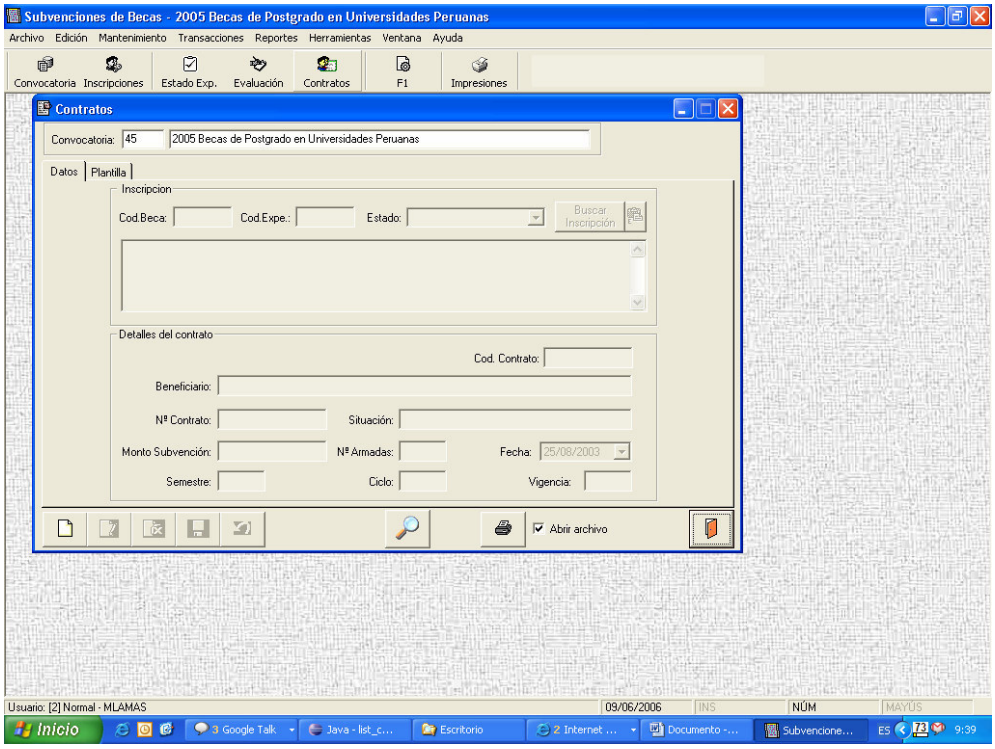

Fig. 17 Sistema actual –Proyectos: contratos

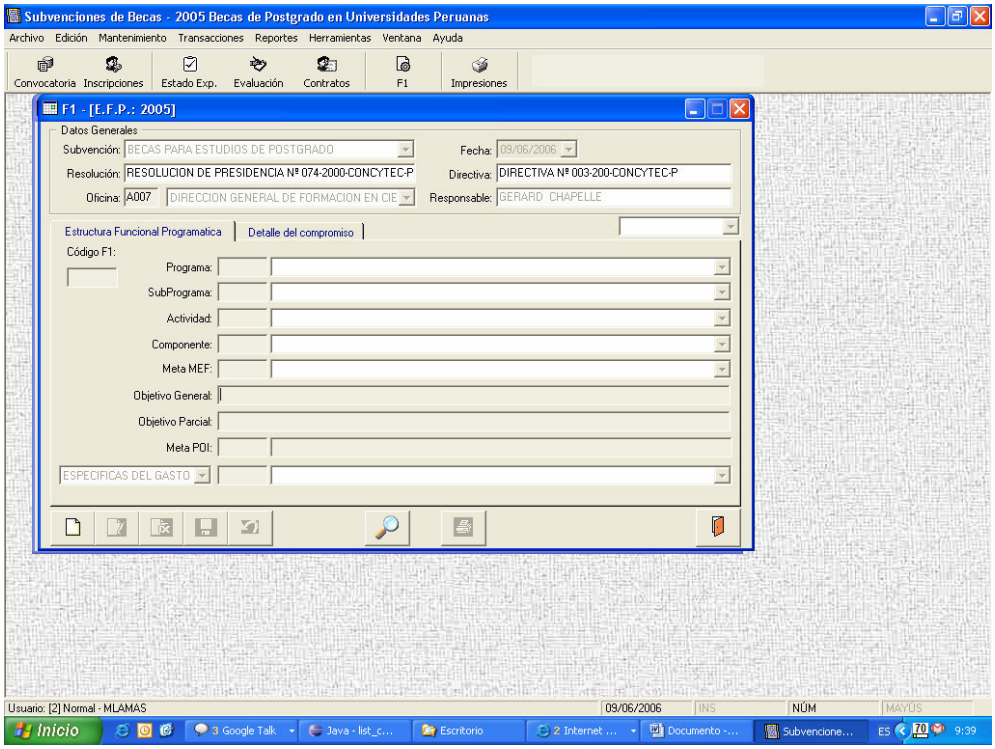

Fig. 18 Sistema actual –Proyectos: generación F1

5.4 Sistema web propuesto: "Sistema de Gestión de Fondos Concursales" SGFC

5.4.1 Características

El sistema propuesto estará soportado bajo una arquitectura Web, bajo la siguiente plataforma de desarrollo:

- Lenguaje de Programación: Java J2EE
- Base de Datos: Postgresql
- Servidor de Aplicaciones: Jboss
- Ide: Eclipse con pluggin My Eclipse
- Control de Versiones: CVS del Visual Source Safe
- FrameWork: JSF

5.4.2 Módulos

La aplicación estará constituida por los siguientes módulos:

- Módulo de Información general
- Módulo Postulación y Evaluación
- Módulo de Contratos y Seguimiento

5.4.3 Plataforma del SGFC

Para administrar de una forma eficiente la información que se genera en estos procesos de subvenciones, se requiere de una plataforma que de todas las facilidades a los agentes involucrados tales como solicitantes, subvencionados, evaluadores pares, investigadores, administrativos del CONCYTEC entre otros.

Para ello la plataforma deberá disponer de sistemas que sean accesibles en algunos de sus procesos a traves de internet, tales como la postulación, la evaluación y el seguimiento. A nivel administrativo deberá de soportar los procesos de evaluaciones, clasificación, contratos, y los demás procesos administrativos.

En la siguiente figura podemos observar la plataforma necesaria para estos sistemas.

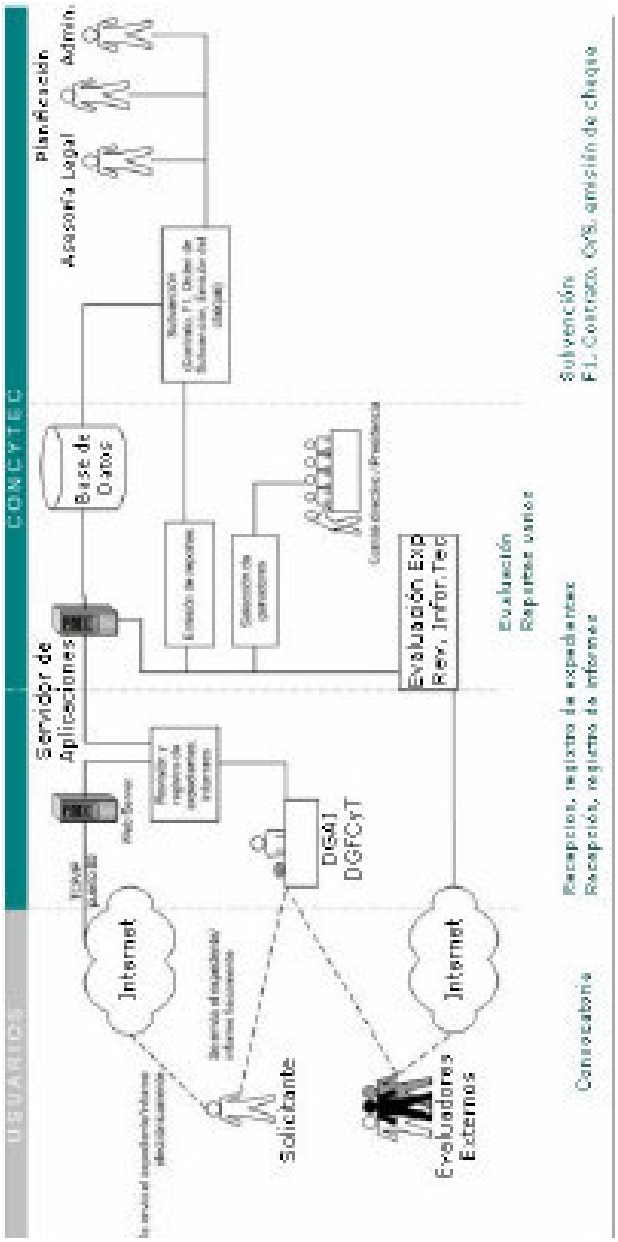

Fig. 19 Plataforma de información propuesta para el SGFC

5.4.4 Modelo de casos de uso del negocio/sistema

El SGFC se desarrolló usando la metodología RUP y para el modelado del negocio y del sistema se usó el lenguaje UML. A continuación se muestra parte del modelado tanto de los casos de uso del negocio como del sistema.

- **Módulo Información general: casos de uso del negocio**

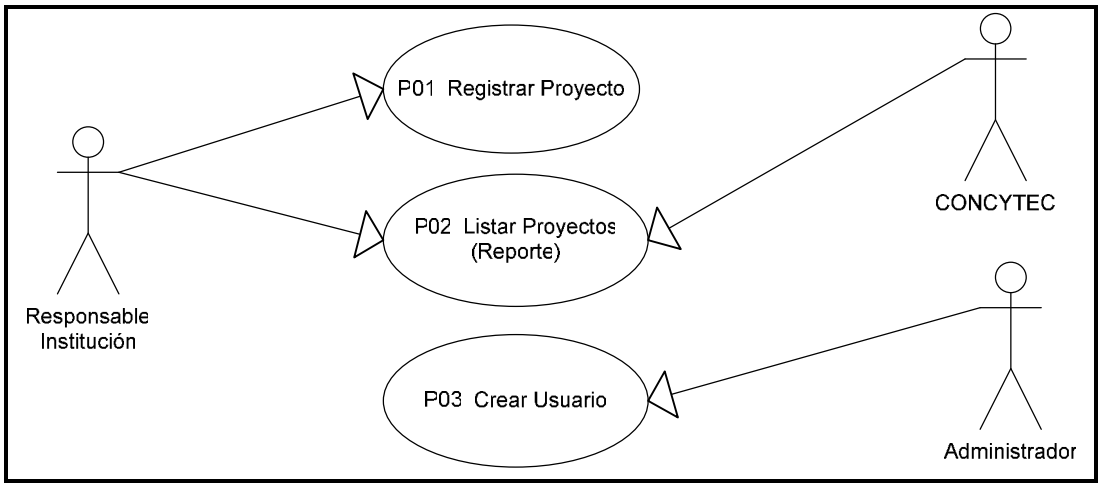

Fig. 20 Diagrama casos de uso del negocio módulo Información personal

- **Módulo Postulación y Evaluación: casos de uso del negocio**
  - Convocatoria

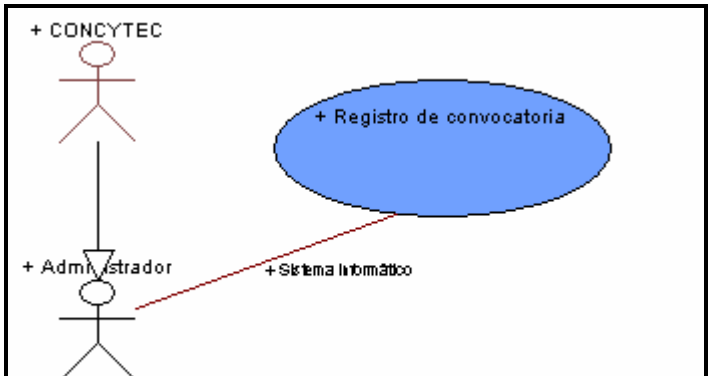

Fig. 21 Diagrama 1 casos de uso módulo Postulación

- Inscripción

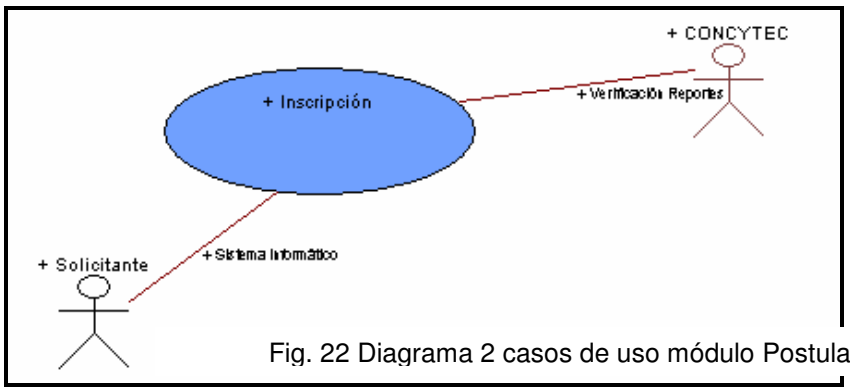

Fig. 22 Diagrama 2 casos de uso módulo Postulación

- Cambio de Estado

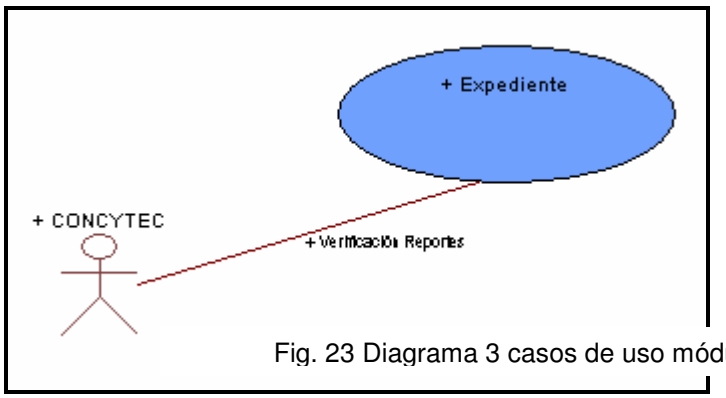

Fig. 23 Diagrama 3 casos de uso módulo Postulación

- Registro de tablas

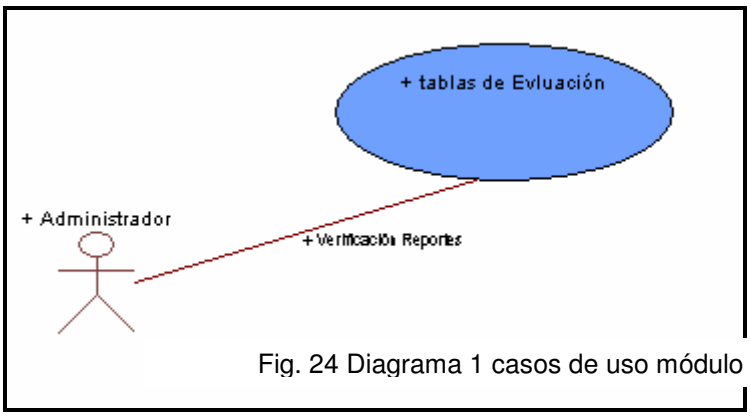

Fig. 24 Diagrama 1 casos de uso módulo Evaluación

- Evaluación

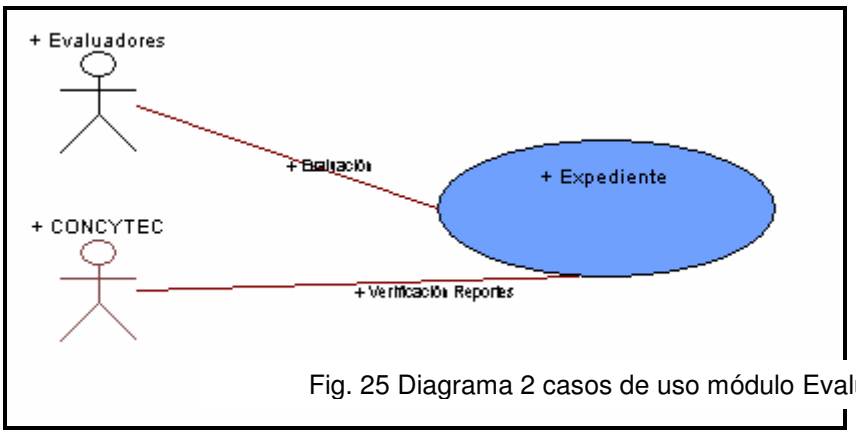

Fig. 25 Diagrama 2 casos de uso módulo Evaluación

- **Módulo Contratación y Seguimiento: casos de uso del sistema**

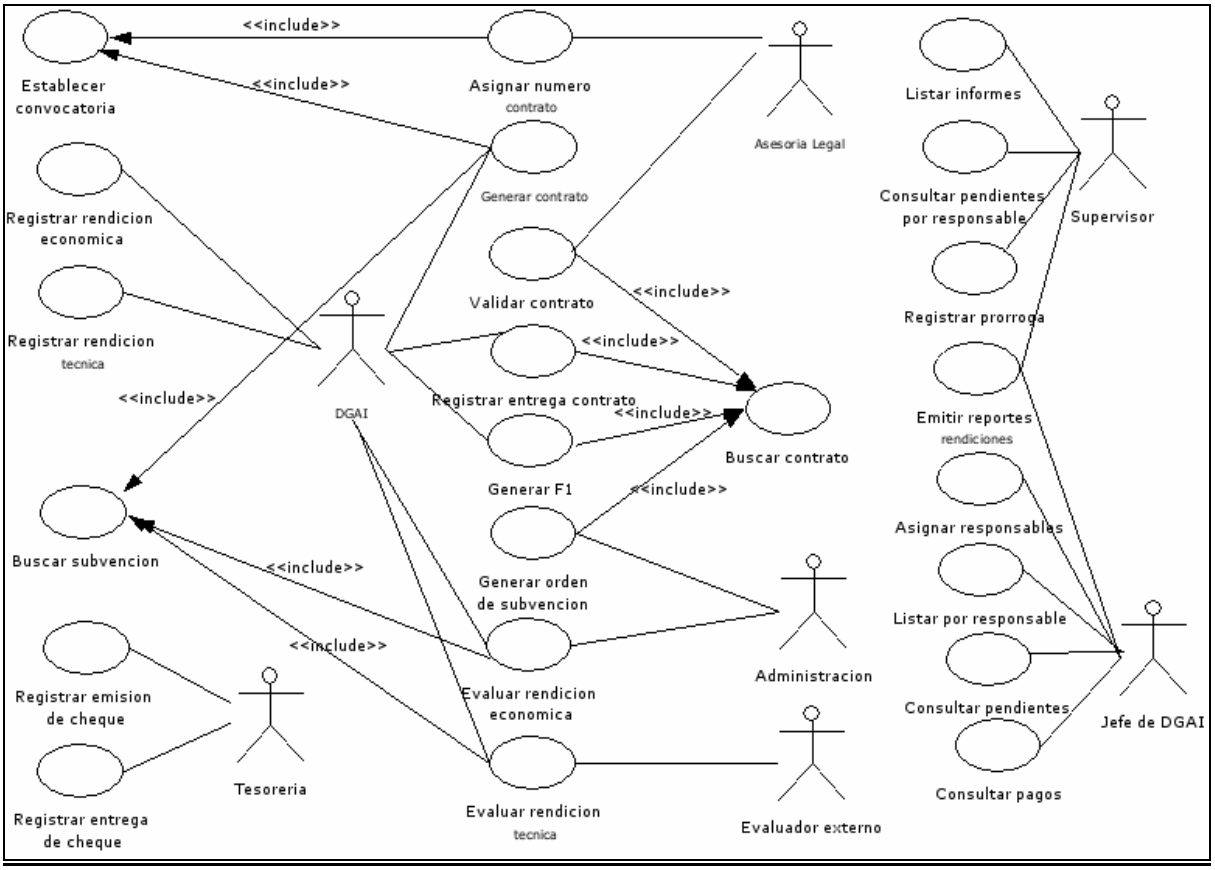

Fig. 26 Diagrama casos de uso del sistema módulo Contratos y Seguimiento

5.4.5 Arquitectura de software

Es este apartado se describe la arquitectura propuesta para la construcción del Sistema de Gestión de Fondos Concursales basado en tecnología java bajo la plataforma JSF. El objetivo conseguir un producto sólido, estructurado y, sobre todo, de fácil mantenimiento.

Descomposición funcional del sistema

Uno de los objetivos más claros e importantes en el proceso de desarrollo de software ha de ser la reutilización de código. Este principio hay que contemplarlo desde dos perspectivas distintas: no desarrollar elementos o componentes que ya existan y estén probados, y desarrollar pensando en que puede que mañana el código que se está implementando pueda ser reutilizado en distintos proyectos. El factor de reutilización de un paquete de código viene determinado por la independencia que exista entre los componentes del sistema. Si una clase realiza varias invocaciones directas a otra distinta, no será portable a otro proyecto a no ser que no llevemos la segunda también. A un nivel más alto de abstracción se debe también procurar desarrollar módulos lo más independientes posible del resto de la aplicación.

Separación Lógica en capas

A la hora de plantear el diseño de una aplicación web, el primer paso es conseguir separar conceptualmente las tareas que el sistema debe desempeñar entre las distintas capas lógicas y en base a la naturaleza de tales tareas. Se ha de partir de la separación inicial en tres

capas, diferenciando que proceso de los que hay que modelar responde a tareas de presentación, cual a negocio y cual a acceso a datos. En caso de identificar algún proceso lógico que abarque responsabilidades adjudicadas a dos o más capas distintas, es probable que dicho proceso deba ser explotado en subprocesos iterativamente, hasta alcanzar el punto en el que no exista ninguno que abarque más de una capa lógica.

a) Capa de presentación

Como se dijo anteriormente, es la responsable de todos los aspectos relacionados con la interfaz de usuario de la aplicación. Así, en esta capa de resuelven cuestiones como:

- Navegabilidad del sistema, mapa de navegación, etc
- Formateo de los datos de salida: Resolución del formato más adecuado para la presentación de resultados. Está relacionado directamente con la internacionalización de la aplicación
- . Internacionalización: Los textos, etiquetas, y datos en general a presentar se obtendrán de uno u otro fichero de recursos en base al idioma preferido del navegador del usuario. En base a esta condición se ven afectadas las representaciones numéricas, las validaciones sobre los datos de entrada (coma decimal o punto decimal) y otros aspectos relativos al idioma del usuario remoto.
- Validación de los datos de entrada, en cuanto a formatos, longitudes máximas, etc.
- . Interfaz gráfica con el usuario.

- . Multicanalidad de la aplicación: Una misma aplicación web puede contar con varias presentaciones distintas, determinándose el uso de la adecuada en base al dispositivo visualizador desde el que trabaje el usuario. Así, no se representará la misma información con el mismo formato en un navegador web estándar que en un dispositivo móvil provisto de un navegador WAP.

Esta es la capa donde interviene de forma directa el framework JSF. Utilizando las etiquetas JSF se mostrarán diversos componentes en las vistas, los que permitirán obtener atractivas interfaces de usuario.

b) Capa manejadora de las vistas

Esta es una capa que se desprende netamente de la utilización del framework JSF, es decir cada vista tendrá un bean manejador , el cual será el intermediario entre la vista y el modelo de datos. Cada componente de la vista se asocia a una propiedad del bean, a su vez otros componentes como los botones se asocian a algún método del bean. De esta manera los beans manejadores serán los encargados de actualizar el modelo de datos.

c) Capa de negocio

En esta capa es donde se deben implementar todas aquellas reglas obtenidas a partir del análisis funcional del proyecto. Así

mismo, debe ser completamente independiente de cualquiera de los aspectos relacionados con la presentación de la misma. De esta forma, la misma capa de negocio debe poder ser empleada para una aplicación web común o una aplicación WAP. Por otro lado, la capa de negocio ha de ser también completamente independiente de los mecanismos de persistencia empleados en la capa de acceso a datos. Cuando la capa de negocio requiera recuperar o persistir entidades o cualquier conjunto de información, lo hará siempre apoyándose en los servicios que ofrezca la capa de acceso a datos para ello. De esta forma, la sustitución del motor de persistencia no afecta lo más mínimo a esta parte del sistema. Las responsabilidades que conviene abordar en esta capa son:

- . Implementación de los procesos de negocio identificados en el análisis del proyecto.

  Como se deduce del párrafo anterior, los procesos de negocio implementados en esta capa son totalmente independientes de cualquier aspecto relativo a la presentación de los mismos. Pongamos por ejemplo la generación de un informe que conste de varias filas las cuales deberán sombrearse con un color determinado por la superación o no de ciertos umbrales para ciertos valores. La aplicación de ciertos umbrales, y la consecuente determinación de la situación (correcta, incorrecta, etc) de cada valor de la fila es un cálculo de negocio que debe afrontarse en esta capa. Sin embargo, sería en un error completar un campo adicional en el que, como

resultado del cálculo, se determinara directamente el color de la fila. Lo adecuado es codificar la situación de la misma y, una vez en presentación, determinar el color adecuado en base al código recibido.

De esta forma, si el usuario requiere que a partir de cierta fecha, el color rojo sea sustituido por el naranja, la modificación de este aspecto de presentación de información se limitaría en la aplicación a una modificación de la capa de presentación.

- . Control de acceso a los servicios de negocio.

Dado que una misma aplicación puede contar con más de una capa de presentación al mismo tiempo, es aconsejable que la responsable última de ejecutar tareas sobre el control de acceso a los servicios del sistema no sea la capa de presentación, sino la de negocio. Los implementadores de la capa de negocio ni pueden ni deben confiar en que las futuras implementaciones de nuevas capas de presentación gestionen adecuadamente el acceso a los servicios de negocio. De esta forma, en cada invocación a cualquier método restringido de negocio se deberá comprobar por medio del sistema de autentificación adecuado, los derechos del usuario actual a realizar tal operación.

- . Publicación de servicios de negocio

Tal y como se explicó en el apartado relativo a la descomposición funcional del sistema al hablar de las microaplicaciones, el lugar adecuado para que dos microaplicaciones o aplicaciones completas interactúen es a nivel de la capa de negocio.

- . Invocación a la capa de persistencia

Los procesos de negocio son los que determinan que, como y cuando se debe persistir en el repositorio de información. Los servicios ofertados por la interfaz de la capa de acceso a datos son invocados desde la capa de negocio en base a los requerimientos de los procesos en ella implementados.

Ésta capa estará formada por un conjunto de clases java, los manejadores, que serán las encargados de aplicar la lógica del negocio. Interectuarán de forma directa con la capa de acceso a datos.

d) Capa de acceso a datos

La capa de acceso a datos es la responsable de la gestión de la persistencia de la información manejada en las capas superiores. Si atendemos a una aplicación web común implementada en java sobre una base de datos relacional, lo más probable es encontrarse con un modelo arquitectónico de acceso a datos consistente en un conjunto de servicios verticales, uno

por cada una de las entidades presentes en el modelo de datos, que ofrecen a las capas superiores las operaciones de persistencia. Las entidades del dominio están representadas cada una por un javabean, una clase que contiene un atributo y sus correspondientes métodos get y set de acceso por cada miembro de la entidad, y que tienen la interesante característica de poder ser introspeccionados, de forma que se pueda conocer lo que en última instancia sería la estructura de una tabla en un modelo relacional.

Clases java son las que realmente implementan la lógica de persistencia e invocan al gestor de bases de datos para satisfacer los servicios solicitados desde negocio. Sus métodos ejecutarán las sentencias SQL necesarias (suponiendo un gestor de bases de datos relacionales). Esta es la arquitectura mayormente recomendada por los fabricantes de servidores de aplicaciones (Sun, Resine CMP, Bea Systems, etc) y por los arquitectos más célebres (Martin Fowler, Beck, etc).

En definitiva, lo que se persigue es una capa de persistencia que goce de las siguientes cualidades:

- . La capa de acceso a datos encapsula toda la lógica de almacenamiento, independizando al resto del sistema del mecanismo de persistencia. Si en un momento dado, tuviéramos que sustituir la habitual base de datos relacional por una base de datos orientada a objetos, o por

simples ficheros XML, el resto de la aplicación no sufriría impacto alguno.

Para esta capa se utilizará un conjunto de clases java, los despachadores, los que serán los encargados de realizar las consultas a la base de datos.

e) Capa de consultas

Adicionalmente se utilizará otra capa, formada por un conjunto de clases java que contendrán las sentencias SQL, esto facilita la migración de una base de datos a otra, dado que el impacto se limitaría a configuración, sin necesidad de codificación, compilación, reempaquetamiento o despliegue.

5.4.6 Interfaces de usuario

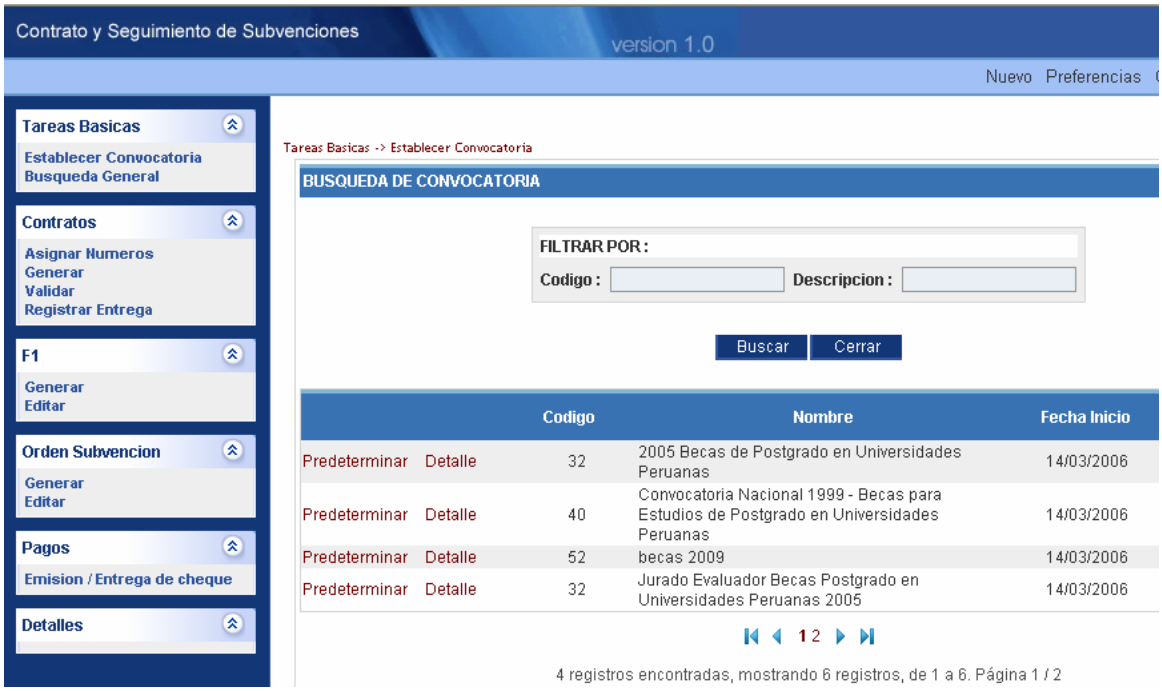

Fig. 27 Sistema nuevo convocatoria

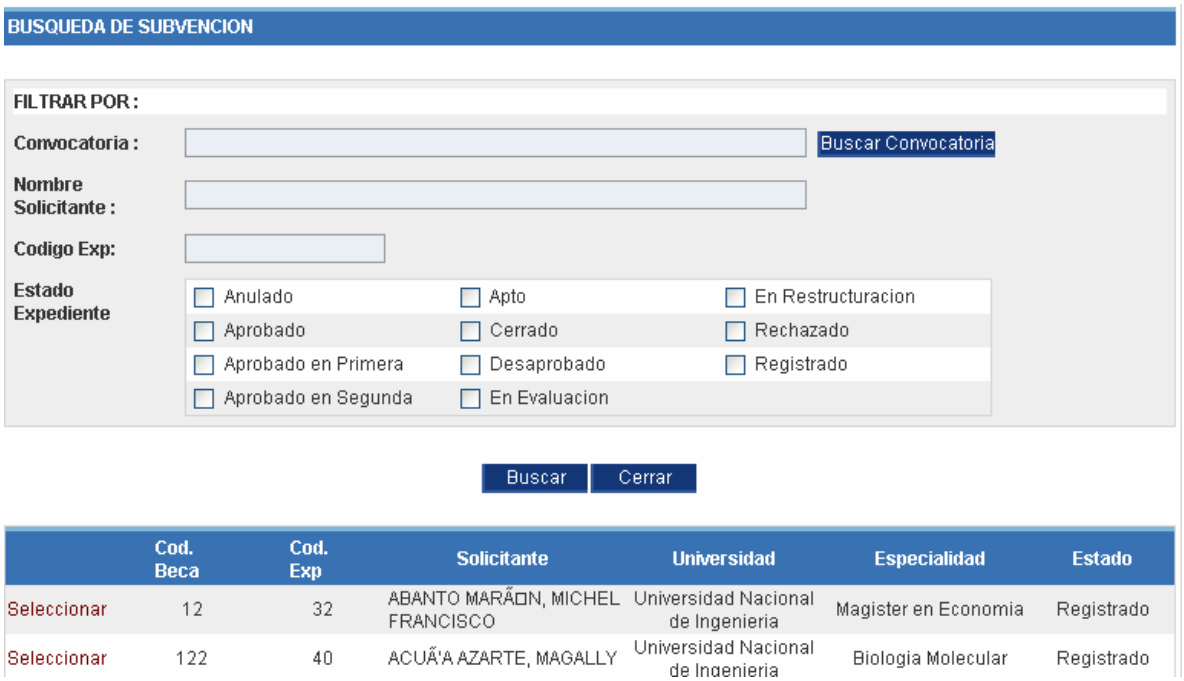

Fig. 28 Sistema nuevo búsqueda de subvención

Fig. 29 Sistema nuevo –Registrar contrato

Fig. 30 Sistema nuevo Listado de pagos

5.5 Comparación entre el sistema actual y el sistema nuevo (SGFC)

5.5.1 Distribución módulos

En el sistema actual el sistema se divide en módulos de acuerdo al tipo de subvención: becas, proyectos, tesis, eventos, publicaciones, premios; es decir se desarrollan las mismas vistas seis veces, como se aprecian en la sección 5.3.4 donde se muestran algunas vistas del módulo de becas y algunas vistas del módulo proyectos.

En el sistema nuevo el sistema se divide en módulos de acuerdo a los procesos involucrados, así tenemos los siguientes módulos: Información general, Postulación y Evaluación, Contratación y Seguimiento, en cada una de las pantallas se podrá realizar las tareas de cualquier tipo de subvención.

5.5.2 Ventajas a obtener al optar por el nuevo sistema

- Información actualizada y confiable de la data referente a las subvenciones, la que permitirá obtener por ejemplo listados actualizados de los subvencionados deudores, relación de los informes presentados y pendientes de los subvencionados, y de esta manera realizar un seguimiento efectivo a los subvencionados.

- Eliminación de las largas colas en las oficinas del Concytec, al implementar una solución web, se descentralizan ciertos procesos críticos como son las inscripción de los postulantes, la que

usualmente demoraba de 35 a 40 minutos se reduce a 7 minutos, ya que se reduce a verificar documentos adjuntos y hacer el ingreso de mínima cantidad de datos puesto que la mayoría ha sido ingresada previamente por el postulante vía web.

- Integración de todas las dependencias involucradas en el proceso de entrega de subvenciones, se eliminaran los registros realizados en hojas de cálculos aisladas del sistema actual que realizaba la oficina de Administración, para efectos de registrar las rendiciones.

CAPITULO VI

6.- CONCLUSIONES

6.1 Sobre el sistema de subvenciones:

- La etapa de seguimiento de una subvención a través del "*Sistema de Gestión de Fondos Concursales"* SGFC, se redujo en en un 60% (Fuente: usuarios de DGAI y Administración), ya que no deben revisar las hojas de cálculos aisladas y constrastarlas, en su lugar son consultadas a través del SGFC, para cada área independientemente cuando se requiera.
- En el *"Sistema de subvenciones"* (de escritorio) al orientar los módulos en función del tipo de subvención (Becas, Tesis, Proyectos, Eventos, Publicaciones, Premios) se incrementó el número de vistas a desarrollar y por ende el tiempo de desarrollo, se puede deducir entonces que la separación en módulos por tipo de subvención no es la más apropiada, en su lugar se debe separar los módulos en función a los procesos a desarrollar en el aplicativo, que serán los mismos para todo tipo de subvención.
- Al orientar la aplicación a un entorno web nos permite una mayor interacción del subvencionado en el proceso de las subvenciones,

ya que podrá consultar el estado de la subvención en cualquier momento, así como realizar otras acciones como son las rendiciones e incluso la postulación.

6.2 Sobre el framework JSF:

- JSF es una tecnología de interfaces de usuario centrada en el MVC, orientada al desarrollo rápido y en creciente auge.
- JSF permite reutilización, separación de roles, facilidad de uso y reducir el tiempo de desarrollo.
- JSF se puede combinar con otros frameworks para obtener un soporte más potente, por ejemplo con Struts y Spring.
- Entre los beneficios que se obtienen al utilizar el framework JSF tenemos:
    - Soporta el patrón MVC (Modelo-Vista-Controlador).
    - Permite extender los componentes existentes para obtener otros más complejos.
    - Amplio soporte JSF en los principales IDE's para el desarrollo de aplicaciones web java.
    - Facilidad de Uso.
    - Estandarización.
    - Independencia de Plataforma.

CAPITULO VII

7.- RECOMENDACIONES

- Si se tiene una aplicación basada en struts, y se desea usar algunos de los sofisticados componentes de JavaServer Faces para mejorar el interface de usuario de la aplicación pero no se tiene tiempo de reescribir completamente la aplicación para basarla en los APIs de JavaServer Faces, lo más recomendable e utilizar la librería de integración Struts-Faces creada expresamente para este própósito. El objetivo principal de esta librería es permitir a los desarrolladores migrar componentes de interface de usuario de aplicaciones existentes desde Struts a JSF, página a página, con cambios mínimos en los ficheros de configuración existentes (struts-config.xml) y sin cambios en las lógicas de negocio o de persistencia. La librería funciona con Struts 1.1 y con Struts 1.2.x.

CAPITULO VIII

8.- REFERENCIAS

8.1 Principales referencias de libros

GEARY David · HORSTMANN Cay
2004. *Core Java Server Faces.* Prentice Hall

MANN Kito D.
2005. *Java Server Faces in Action.* Mannig

8.2 Principales referencias de páginas web

Ventajas e inconvenientes de las aplicaciones web
http://www.avidos.net/blogold/aplicaciones-web/

CELI, Ismael
2006 *Sobre las ventajas y desventajas de las RIA's.*
http://www.estadobeta.com/2006/05/07/rias/ :7/5/2006

CONCYTEC
http://www.concytec.gob.pe

CORSI, Miguel
2006 *Beneficios De Las Aplicaciones Basadas En Web Y El Anuncio De Microsoft De La Era "En Vivo"*
http://www.masternewmedia.org/es/aplicaciones_web/temas_de_aplicaciones_web/Beneficios_De_Las_Aplicaciones_Basadas_En%20_Web_Y_El_Anuncio_De_Microsoft_De_La_Era_En_Vivo.htm : 01/2006

FERNÁNDEZ LANVÍN, Daniel
Arquitectura web
http://www.di.uniovi.es/~dflanvin/docencia/dasdi/teoria/Transparencias/06.%20Arquitectura%20Web.pdf

FERNÁNDEZ LANVÍN, Daniel
Definición de una arquitectura software para el diseño de aplicaciones web basadas en tecnología Java-J2EE
http://www.di.uniovi.es/~dflanvin/docencia/dasdi/teoria/Transparencias/06.%20Arquitectura%20Web.pdf

GUERRERO, Luis A.
Modelando aplicaciones Web con UML
http://www.dcc.uchile.cl/~luguerre/cc61j/recursos/web-app.ppt

MATEU, Carlos
2004 *Desarrollo de aplicaciones web*
http://www.uoc.edu/masters/esp/img/873.pdf : /2004

SÁNCHEZ RAMÓN, Óscar
2006: *Java Server Faces El siguiente paso en el desarrollo web*
http://ditec.um.es/ssdd/trabajos/0506/exposicionJSF.ppt#310,54,6. Referencias :05/06

SILVA, Darío Andrés - MERCERAT, Bárbara
2002 *Construyendo aplicaciones web con una metodología de diseño orientada a objetos.*
http://www.unab.edu.co/editorialunab/revistas/rcc/pdfs/r22_art5_c.pdf: 01/2002

WIKIPEDIA
http://es.wikipedia.org/: 07/2006

CAPITULO IX

9.- ANEXOS

9.1 Introducción a Internet

Internet, la red de redes, nace a mediados de la década de los setenta, bajo los auspicios de DARPA, la Agencia de Proyectos Avanzados para la Defensa de Estados Unidos. DARPA inició un programa de investigación de técnicas y tecnologías para unir diversas redes de conmutación de paquetes, permitiendo así a los ordenadores conectados a estas redes comunicarse entre sí de forma fácil y transparente.

De estos proyectos nació un protocolo de comunicaciones de datos, IP o Internet Protocol, que permitía a ordenadores diversos comunicarse a través de una red, Internet, formada por la interconexión de diversas redes. A mediados de los ochenta la Fundación Nacional para la Ciencia norteamericana, la NSF, creó una red, la NSFNET, que se convirtió en el backbone (el troncal) de Internet junto con otras redes similares creadas por la NASA (NSINet) y el U.S. DoE (Department of Energy) con la ESNET. En Europa, la mayoría de países disponían de backbones nacionales (NORDUNET, RedIRIS, SWITCH, etc.) y de una serie de iniciativas paneuropeas (EARN y RARE). En esta época

aparecen los primeros proveedores de acceso a Internet privados que ofrecen acceso pagado a Internet.

A partir de esta época, gracias entre otras cosas a la amplia disponibilidad de implementaciones de la suite de protocolos TCP/IP (formada por todos los protocolos de Internet y no sólo por TCP e IP), algunas de las cuales eran ya de código libre, Internet empezó lo que posteriormente se convertiría en una de sus características fundamentales, un ritmo de crecimiento exponencial, hasta que a mediados del 2002 empieza a descender ligeramente el ritmo de crecimiento.

A mediados de los noventa se inició el boom de Internet. En esa época el número de proveedores de acceso privado se disparó, permitiendo a millones de personas acceder a Internet, que a partir de ese momento ya se empezó a conocer como la Red, desbancado a las demás redes de comunicación existentes (Compuserve, FidoNet/BBS, etc.). El punto de inflexión vino marcado por la aparición de implementaciones de TCP/IP gratuitas (incluso de implementaciones que formaban parte del sistema operativo) así como por la popularización y abaratamiento de medios de acceso cada vez más rápidos (módems de mayor velocidad, RDSI, ADSL, cable, satélite). El efecto de todos estos cambios fue de “bola de nieve”: a medida que se conectaban más usuarios, los costes se reducían, aparecían más proveedores e Internet se hacía más atractivo y económico, con lo que se conectaban más usuarios, etc.

En estos momentos disponer de una dirección de correo electrónico, de acceso a la web, etc., ha dejado de ser una novedad para convertirse en algo normal en muchos países del mundo. Por eso las empresas, instituciones, administraciones y demás están migrando rápidamente todos sus servicios, aplicaciones, tiendas, etc., a un entorno web que permita a sus clientes y usuarios acceder a todo ello por Internet. A pesar del ligero descenso experimentado en el ritmo de crecimiento, Internet está destinado a convertirse en una suerte de servicio universal de comunicaciones, permitiendo una comunicación universal.

9.2 La WWW como servicio de Internet

La WWW (World Wide Web) o, de forma más coloquial, la web, se ha convertido, junto con el correo electrónico, en el principal caballo de batalla de Internet. Ésta ha dejado de ser una inmensa “biblioteca” de páginas estáticas para convertirse en un servicio que permite acceder a multitud de prestaciones y funciones, así como a infinidad de servicios, programas, tiendas, etc.

9.2.1 Breve historia de la WWW

En 1989, mientras trabajaba en el CERN (Centro Europeo de Investigación Nuclear), Tim Berners-Lee empezó a diseñar un sistema para hacer accesible fácilmente la información del CERN. Dicho sistema empleaba el hipertexto para estructurar una red de enlaces entre los documentos. Una vez obtenida la aprobación para continuar el proyecto, nació el primer navegador web, llamado World- WideWeb (sin espacios). En 1992 el sistema ya se había extendido fuera del CERN. El número de

servidores “estables” había aumentado, alcanzando la sorprendente cifra de veintiséis. A partir de este punto, el crecimiento es espectacular. En 1993 la web ya era merecedora de un espacio en el New York Times. Éste es el año del lanzamiento de Mosaic, un navegador para X-Window/ Unix que con el tiempo se convertiría en Netscape y que fue un factor clave de popularización de la web. En 1994 se fundó el WWW Consortium, que se convertiría en el motor de desarrollo de los estándares predominantes en la web (http://www.w3c.org). A partir de ese momento, el crecimiento ya fue constante, convirtiéndose hacia finales de los noventa en el servicio insignia de Internet y dando lugar al crecimiento imparable de los servicios en línea que estamos experimentado actualmente.

9.2.2 Fundamentos de la web

El éxito espectacular de la web se basa en dos puntales fundamentales: el protocolo HTTP y el lenguaje HTML. Uno permite una implementación simple y sencilla de un sistema de comunicaciones que nos permite enviar cualquier tipo de ficheros de una forma fácil, simplificando el funcionamiento del servidor y permitiendo que servidores poco potentes atiendan miles de peticiones y reduzcan los costes de despliegue. El otro nos proporciona un mecanismo de composición de páginas enlazadas simple y fácil, altamente eficiente y de uso muy simple.

a) El protocolo HTTP

El protocolo HTTP (hypertext tranfer protocol) es el protocolo base de la WWW. Se trata de un protocolo simple, orientado a conexión y sin estado. La razón de que esté orientado a conexión es que emplea para su funcionamiento un protocolo de comunicaciones (TCP, transport control protocol) de modo conectado, un protocolo que establece un canal de comunicaciones de extremo a extremo (entre el cliente y el servidor) por el que pasa el flujo de bytes que constituyen los datos que hay que transferir, en contraposición a los protocolos de datagrama o no orientados a conexión que dividen los datos en pequeños paquetes (datagramas) y los envían, pudiendo llegar por vías diferentes del servidor al cliente. El protocolo no mantiene estado, es decir, cada transferencia de datos es una conexión independiente de la anterior, sin relación alguna entre ellas, hasta el punto de que para transferir una página web tenemos que enviar el código HTML del texto, así como las imágenes que la componen, pues en la especificación inicial de HTTP, la 1.0, se abrían y usaban tantas conexiones como componentes tenía la página, trasfiriéndose por cada conexión un componente (el texto de la página o cada una de las imágenes).

Existe una variante de HTTP llamada HTTPS (S por secure) que utiliza el protocolo de seguridad SSL (secure socket layer) para cifrar y autenticar el tráfico entre cliente y servidor, siendo ésta muy usada por los servidores web de comercio electrónico, así como por aquellos que contienen información personal o confidencial. De manera esquemática,

el funcionamiento de HTTP es el siguiente: el cliente establece una conexión TCP hacia el servidor, hacia el puerto HTTP (o el indicado en la dirección de conexión), envía un comando HTTP de petición de un recurso (junto con algunas cabeceras informativas) y por la misma conexión el servidor responde con los datos solicitados y con algunas cabeceras informativas.

El protocolo define además cómo codificar el paso de parámetros entre páginas, el tunelizar las conexiones (para sistemas de firewall), define la existencia de servidores intermedios de cache, etc. Las directivas de petición de información que define HTTP 1.1 (la versión considerada estable y al uso) son:

GET Petición de recurso.

POST Petición de recurso pasando parámetros.

HEAD Petición de datos sobre recurso.

PUT Creación o envío de recurso.

DELETE Eliminación de recurso.

TRACE Devuelve al origen la petición tal como se ha recibido en el receptor, para depurar errores.

OPTIONS Sirve para comprobar las capacidades del servidor.

CONNECT Reservado para uso en servidores intermedios capaces de funcionar como túneles.

b) El lenguaje HTML

El otro puntal del éxito del WWW ha sido el lenguaje HTML (hypertext mark-up language). Se trata de un lenguaje de marcas (se utiliza insertando marcas en el interior del texto) que nos permite representar de forma rica el contenido y también referenciar otros recursos (imágenes, etc.), enlaces a otros documentos (la característica más destacada del WWW), mostrar formularios para posteriormente procesarlos, etc.

El lenguaje HTML actualmente se encuentra en la versión 4.01 y empieza a proporcionar funcionalidades más avanzadas para crear páginas más ricas en contenido. Además se ha definido una especificación compatible con HTML, el XHTML (extensible hypertext markup language) que se suele definir como una versión XML validable de HTML, proporcionándonos un XML Schema contra el que validar el documento para comprobar si está bien formado, etc.

9.3 Características de un servidor web

9.3.1 Servicio de ficheros estáticos

Todos los servidores web deben incluir, como mínimo, la capacidad para servir los ficheros estáticos que se encuentren en alguna parte concreta del disco.

9.3.2 Seguridad y autenticación

El modo más simple de control es el proporcionado por el uso de ficheros .htaccess. Éste es un sistema de seguridad que proviene de uno de los primeros servidores web (del NCSA, *National Certer for Supercomputing Applications*, Centro Nacional de Aplicaciones de Supercomputación), que consiste en poner un fichero de nombre .htaccess en cualquier directorio del contenido web que se vaya a servir, indicando en este fichero qué usuarios, máquinas, etc., tienen acceso a los ficheros y subdirectorios del directorio donde está el fichero.

Otros servidores permiten especificar reglas de servicio de directorios y ficheros en la configuración del servidor web, indicando allí qué usuarios, máquinas, etc., pueden acceder al recurso indicado.

Por lo que respecta a la autenticación (validación del nombre de usuario y contraseña proporcionados por el cliente), la mayoría permiten, como mínimo, proporcionar al servidor web un fichero con nombres de usuario y contraseñas contra el que se pueda validar lo enviado por el cliente.

9.3.4 Contenido dinámico

Uno de los aspectos más importantes del servidor web escogido es el nivel de soporte que nos ofrece para servir contenido dinámico. Dado que la mayor parte del contenido web que se sirve se genera dinámicamente, el soporte para contenido dinámico que nos ofrece el servidor web es uno de los puntos más críticos en su elección. La mayoría de servidores web ofrecen soporte para CGI. Muchos ofrecen

soporte para algunos lenguajes de programación (básicamente interpretados) como PHP, JSP, ASP, Pike, etc.

9.3.5 Servidores virtuales

Una prestación que está ganado adeptos y usuarios a marchas forzadas, especialmente entre los proveedores de servicios de Internet y los operadores de alojamiento de dominios, es la capacidad de algunos servidores web de proporcionar múltiples dominios con sólo una dirección IP, discriminando entre los diversos dominios alojados por el nombre de dominio enviado en la cabecera de la petición HTTP. Esta prestación permite administrar de una forma más racional y ahorrativa un bien escaso, como son las direcciones IP.

9.3.6 Actuación como representantes

Algunos servidores web nos permiten usarlos como servidores intermedios (proxy servers). Podemos usar los servidores intermedios para propósitos muy variados:

• Para servir de aceleradores de navegación de nuestros usuarios (uso como proxy-cache).

• Para servir como aceleradores de acceso frontales para un servidor web, instalando diversos servidores web que repliquen los accesos a un servidor maestro (reverse-proxy o HTTP Server acceleration).

• Como frontales a algún servidor o protocolo.

9.3.7 Protocolos adicionales

Algunos servidores, además de atender y servir peticiones HTTP (y HTTPS), pueden atender y servir peticiones de otros protocolos o de protocolos implementados sobre HTTP. Algunos de estos protocolos pueden convertirse en requisitos fundamentales de nuestro sistema y, por ello, su existencia en el del servidor web puede ser imprescindible.

9.4 Niveles de una aplicación web

– **El nivel de interfaz de usuario** está compuesto por las páginas HTML que el usuario solicita a un servidor web y que visualiza en un cliente web (normalmente, un navegador web).

– **El nivel de lógica de negocio** está compuesto por los módulos que implementan la lógica de la aplicación y que se ejecutan en un servidor de aplicaciones.

– **El nivel de datos** está compuesto por los datos, normalmente gestionados por un sistema de gestión de bases de datos (servidor de datos), que maneja la aplicación web.

9.5 Evolución de las aplicaciones web

La evolución tecnológica que el sector ha sufrido durante los últimos años ha permitido otra evolución paralela, la de la arquitectura de las aplicaciones web. A medida que aparecían nuevos recursos técnicos, los patrones de diseño se amoldaban para aprovechar las nuevas característica que estás novedades ofrecían. De esta forma, el modelo arquitectónico de las aplicaciones de Internet ha sufrido dos grandes saltos desde la aparición de los

primeros portales. Los distintos modelos de aplicación sobre los que ha ido desarrollando se clasifica en:

9.5.1 Modelo 1

Son las más primitivas. Se identifican con este modelo las clásicas aplicaciones web CGI, basadas en la ejecución de procesos externos al servidor web, cuya salida por pantalla era el html que el navegador recibía en respuesta a su petición. Presentación, negocio y acceso a datos se confundían en un mismo script perl.

9.5.2 Modelo 1.5

Aplicado a la tecnología java, se da con la aparición de las jsps y los servlets. En este modelo, las responsabilidades de presentación (navegabilidad, visualización, etc) recaen en las páginas jsps, mientras que los beans incrustados en las mismas son los responsables del modelo de negocio y acceso a datos.

9.5.3 Modelo 2

Como evolución del modelo 1.5 con la incorporación del patrón MVC a este tipo de aplicaciones, se define lo que se conoce como Model 2 de la arquitectura web. Se realiza la incorporación de un elemento controlador de la navegación de la aplicación. El modelo de negocio queda encapsulado en los javabeans que se incrustan en las jsps, aunque esta responsabilidad se explota cuando se alcanza el modelo de diseño ncapas .

9.5.4 Modelo 2X

El modelo 2X aparece como evolución del modelo 2, y con objeto de dar respuesta a la necesidad, cada vez más habitual, de desarrollar aplicaciones multicanal, es decir, aplicaciones web pueden ser atacadas desde distintos tipos de clientes remotos. Así, una aplicación web multicanal podrá ejecutarse desde una PDA, desde un terminal de telefonía móvil, o desde cualquier navegador html estándar. El medio para lograr publicar la misma aplicación para distintos dispositivos es emplear plantillas XSL para transformar los datos XML y determinar la plantilla a emplear en base al parámetro user-agent recibido en la request. La aplicación de esta solución al modelo 2 de aplicaciones web define lo que se conoce como modelo 2X.

9.6 Arquitectura software

9.6.1 Definición

Una Arquitectura Software, también denominada Arquitectura lógica, consiste en un conjunto de patrones y abstracciones coherentes que proporcionan el marco de referencia necesario para guiar la construcción del software para un sistema de información. La arquitectura software establece los fundamentos para que analistas, diseñadores, programadores, etc. trabajen en una línea común que permita alcanzar los objetivos y necesidades del sistema de información.

9.6.2 Aspectos generales

A la hora de abordar el desarrollo de un proyecto web, hay una serie de consideraciones acerca del mismo que hay que tener muy presentes, dado que son claves en que tipo de diseño y metodologías de desarrollo aplicar.

- **Escalabilidad**

Una de las características principales de las aplicaciones publicadas en la Web es que el número de usuarios de la misma puede verse incrementado de forma vertiginosa en un periodo de tiempo relativamente corto. El éxito o el fracaso de un sitio web orientado al usuario común determinarán, entre otros aspectos, el dimensionamiento del sistema sobre el que se instala y soporta el software que sustenta dicho sitio. En consecuencia, una de los requisitos fundamentales de una aplicación web es que sea completamente escalable sin que un aumento de los recursos dedicados a la misma suponga modificación alguna en su comportamiento o capacidades. La escalabilidad de un sistema web puede ser:

a) Horizontal: cuando literalmente clonamos el sistema en otra máquina (u otras máquinas) de características similares y balanceamos la carga de trabajo mediante un dispositivo externo. El balanceador de carga puede ser:

Balanceador Software – Por ejemplo, habitualmente encontramos un servidor web apache junto con el módulo mod_jk que permite la redirección de las peticiones http que a tal efecto

sean configuradas entre las distintas máquinas que forman la granja de servidores. Este tipo de balanceadores examinan el paquete http e identifican la sesión del usuario, guardando registro de cual de las máquinas de la granja se está encargado de servir a dicha sesión. Este aspecto es importante, dado que nos permite trabajar (de cara al diseño de la aplicación) apoyándonos en el objeto session propio del usuario y almacenando en ella información relativa a la sesión del mismo, puesto que tenemos la garantía de que todas las peticiones de una misma sesión http van a ser redireccionadas hacia la misma máquina.

Balanceador hardware – Se trata de dispositivos que, respondiendo únicamente a algoritmos de reparto de carga (Round Robin, LRU, etc.), redireccionan una petición http del usuario a la máquina que, según dicho algoritmo, convenga que se haga cargo de la petición. Son mucho más rápidos de los anteriores, dado que se basan en conmutación de circuitos y no examinan ni interpretan el paquete http. Sin embargo, el no garantizar el mantenimiento de la misma sesión de usuario en la misma máquina, condiciona seriamente el diseño, dado que fuerza a que la información relativa a la sesión del usuario sea almacenada por el implementador del mismo, bien en cookies o bien en base de datos.

Balanceador hardware http- Se trata de dispositivos hardware pero que si examinan el paquete http y mantienen la

relación usuario – máquina servidora. Mucho más rápidos que los balanceadores software, pero algo menos que los hardware, suponen hoy en día una de las soluciones más aceptadas en el mercado.

b) Vertical: Habitualmente, la separación lógica en capas se implementa de tal forma que se permita una separación física de las mismas. Interponiendo elementos conectores que actúen de middlewares (por ejemplo, EJBs), es posible distribuir la aplicación de forma vertical (una máquina por cada capa del sistema), e incluso si esto no fuera suficiente, distribuyendo los elementos de una misma capa entre distintas máquinas servidoras.

c) Cluster: Con la aparición de los servidores de aplicaciones en cluster se abre una nueva capacidad de escalabilidad que, dependiendo de cómo se aplique, podría clasificarse como vertical u horizontal. Un cluster de servidores de aplicaciones permite el despliegue de una aplicación web corriente de forma que su carga de trabajo vaya a ser distribuida entre la granja de servidores que forman el cluster, de modo transparente al usuario y al administrador. El cluster, mediante el mecanismo de replicación de sesión, garantiza que sea cual sea la máquina que sirva la petición http tendrá acceso a la sesión del usuario (objeto HttpSession en java). Este tipo de sistemas, debido precisamente

a la replicación de sesión, suele presentar problemas de rendimiento.

Pese a que hace un tiempo la tendencia de diseño era contraria al empleo de la sesión (objeto session) como soporte de apoyo en el desarrollo del sistema, sobre todo debido a problemas de rendimiento y a la abundancia de balanceadores hardware comunes, hoy en día es habitual el empleo de la misma. No obstante, es un recurso delicado, dado que un abuso de está técnica acarrea problemas de rendimiento por un excesivo uso de memoria. Hay que tener en cuenta a la hora de hacer el diseño de una aplicación web java que, hasta que la sesión no caduque, todos los objetos contenidos en la misma que no hayan sido eliminados explícitamente persisten en la memoria del servidor, puesto que no están disponibles para el recolector de basura.

- **Separación de responsabilidades o incumbencias entre los distintos elementos del sistema.**

Esta premisa supone la base de la separación en capas del sistema. Distintas responsabilidades no deben ser delegadas en la misma clase, y llevado esto algo más allá, en el mismo conjunto de clases. En la actualidad, la tendencia más aceptada es la aplicación de patrones de diseño de arquitectura que dividen la responsabilidad en distintas capas (separación de incumbencias) que interaccionan unas con otras a través de sus interfaces. Se trata de los sistemas

denominados multicapa o n-capas. Aplicados a los proyectos web, el modelo más básico es el de aplicaciones 3 capas:

a) Presentación

Como su nombre indica, se limita a la navegabilidad y a gestionar todos aquellos aspectos relacionados con la lógica de presentación de la aplicación, como comprobación de datos de entrada, formatos de salida, internacionalización de la aplicación, etc.

b) Negocio o dominio

El resultado del análisis funcional de la aplicación viene a ser la identificación del conjunto de reglas de negocio que abstraen el problema real a tratar. Estás son las que realmente suponen el motor del sistema, dado que se basan en el funcionamiento del modelo real. En un caso hipotético de un programa de gestión cualquiera, estaríamos hablando, por ejemplo, del conjunto de operaciones que dan el servicio de negocio “emisión de factura”.

c) Acceso a datos

Esta capa es la encargada de persistir las entidades que se manejan en negocio, el acceso a los datos almacenados, la actualización, etc, aunque puede ofrecer servicios relacionados con la persistencia o recuperación de información más complejos.

Tomando como base este modelo, distintos patrones arquitectónicos han ido apareciendo como evolución del mismo (Modelo Brown, los patrones de Sun, etc.), casi todos insertando más capas que habitualmente están orientadas a conseguir una mayor independencia entre las tres anteriormente descritas.

- **Portabilidad**

En la medida de lo posible, una aplicación web debe poder adaptarse a las distintas posibles arquitecturas físicas susceptibles de ser empleadas para el despliegue del paquete, limitándose en la medida de lo posible el impacto de tal adaptación a tareas de configuración, y evitándose así la necesidad de modificar el código de la misma ante dichas situaciones. Un ejemplo que se da habitualmente, es el encontrar un cliente reacio al empleo de tecnologías J2EE por los consabidos costes de rendimiento (o simplemente económicos) que acarrean. Dado que el modelo de separación física de capas mediante Enterprise javabeans de tipo sesión implementando el patrón de diseño Façade es algo habitual y recomendado desde muchos ámbitos académicos, surge la necesidad de modificar el código del producto para eliminar las capas intermedias, puesto que no se cuenta con un contenedor de EJBs para la implantación.

- **Componentización de servicios de infraestructura**

En todas las aplicaciones, incluidas las aplicaciones web, aparecen una serie de servicios que podríamos denominar de

infraestructura, y que han de estar disponibles desde distintas partes del sistema. Así, esta necesidad rompe aparentemente la separación vertical de capas, dando lugar a una capa de infraestructura que sirve funcionalmente a todas las demás. Casos habituales de servicios de esta naturaleza son:

- Servicio de log
- Pool de conexiones JDBC (o de cualquier otro sistema de persistencia)
- Sistema de configuración
- Gestor de accesos/permisos de usuario.
- Etc.

La arquitectura propuesta pretende tratar este conjunto de servicios como componentes, servicios de la capa de infraestructura, de forma que:

-Puedan ser sustituidos por nuevas versiones sin necesidad de parada del sistema ni recompilación y/o repaquetización del mismo

Puedan ser reutilizados por futuros proyectos o distintas partes del mismo, evitando que en un mismo sistema coexistan n distintos gestores de permisos, n distintos proveedores de conexiones a bases de datos, gestores de logs, etc.

-Sean accesibles desde todas las capas del sistema sin romper la independencia entre las mismas.

- **Gestión de la sesión de usuario, cacheado de entidades**

Con objeto de limitar en la medida de lo posible los accesos innecesarios a memoria secundaria (bases de datos, ficheros externos

de configuración, etc) se propone un sistema que se apoya en parte en el empleo de la sesión HTTP(s) para cachear ciertos datos referentes a la sesión del usuario, o bien comunes a todas las sesiones de usuario. Obviamente, la cantidad y naturaleza de las entidades susceptibles de ser cacheadas será determinada teniendo muy presentes aspectos de rendimiento del producto, dado que un empleo abusivo de esta técnica puede y suele llevar a un consumo excesivo de los recursos del sistema (memoria).

- **Aplicación de patrones de diseño**

El empleo y aplicación de patrones de diseño facilita el entendimiento del código y, por tanto, reduce considerablemente el coste de mantenimiento, dado que además de aportar soluciones eficientes para problemas comunes, son muy interesantes como medio de entendimiento entre diseñadores e implementadores.

9.6.3 Importancia del desarrollo en capas

Las antiguas aplicaciones distribuidas en cd's dieron lugar a aplicaciones dinámicas, de constante actualización e incluso personalizables, capaces de adaptarse a los tipos de usuarios y en casos avanzados, a cada usuario en particular. Estas características encuentran el medio ideal en la web, ya que de otra forma sería costoso su mantenimiento y evolución.

La complejidad del desarrollo ocurre a diferentes niveles: dominios de aplicación sofisticados (financieros, médicos, geográficos, etc.); la necesidad de proveer acceso de navegación simple a grandes

cantidades de datos multimediales, y por último la aparición de nuevos dispositivos para los cuales se deben construir interfaces web fáciles de usar. Esta complejidad en los desarrollos de software sólo puede ser alcanzada mediante la separación de los asuntos de modelización en forma clara y modular.

Es importante la clara identificación de los tres diferentes niveles de diseño en forma independiente de la implementación. La justificación de tanto trabajo puede encontrarse en cualquier aplicación que requiera navegación: en términos de programación orientada a objetos, si los elementos por los que se navega son los del diseño conceptual se estaría mezclando la funcionalidad con el comportamiento propio del objeto. Por otro lado, si los nodos de la red de navegación tienen la capacidad de definir su apariencia, se estaría limitando la extensión de la aplicación para ofrecer nuevas presentaciones del mismo elemento y eventualmente se estaría dificultando la personalización de la interfaz.

Es necesario, entonces, mantener separadas las distintas decisiones de diseño según su naturaleza (conceptual, navegacional, de interfaz) y aplicar las tecnologías adecuadas a cada capa en el proceso de implementación.

9.6.4 Patrones de arquitectura web

En el diseño de una aplicación cliente-servidor, hay una decisión que hay que tomar: qué parte de la aplicación debe ser hecha por el

cliente y cuál por el servidor. Esta decisión puede afectar crucialmente el costo del servidor y el cliente, la robustez, la seguridad de toda la aplicación.

a) Arquitectura Cliente liviano.

Un Cliente Liviano (Thin client) es una computadora (cliente) en una arquitectura de red cliente-servidor que tiene muy poca o ninguna lógica del programa, por lo tanto depende principalmente del servidor central para las tareas de procesamiento. La palabra liviano se refiere a lo pequeña que es la imagen de arranque, quiza no más grande que la requerida para conectar a la red y arrancar un navegador web.

b) Arquitectura cliente pesado

En informática, cliente pesado (también conocido como cliente rico o cliente grueso) es un término de arquitectura cliente-servidor para un cliente que realiza el grueso de las operaciones de procesamiento de datos. Los datos en sí mismos son almacenados en el servidor. Aunque el término usualmente se refiere al software, se puede aplicar a una red de PCs que tiene unas habilidades de procesamiento relativamente fuertes.Un ejemplo de cliente pesado sería una aplicación Swing (Java).

A partir del 2003 el término cliente rico (rich client) ha tomado un significado diferente al de cliente pesado. Se ha convertido en un híbrido entre 'liviano' y 'pesado', o una arquitectura

donde el uso del CPU de parte del cliente y del servidor esta más balanceada. El cliente puede parecer y comportarse como un cliente pesado, mientras está completamente basado en una arquitectura de red de cliente liviano.

En la práctica, parece que hay poco donde elegir para decantarse entre una y otra arquitectura para la mayoría de las aplicaciones. Pocas situaciones se decantan claramente hacía una u otra. Los proyectos de computación distribuida (que utilizan una gran cantidad de ordenadores remotos para realizar un análisis computacional intensivo) son aplicaciones que requieren clientes livianos. Por otro lado los sistemas de difusión de entretenimiento multimedia o la difusión de material educativo a muchos clientes puede ser realizada mejor con clientes livianos, ya que se difunde el mismo material a todos los clientes.

CAPITULO X

10.- GLOSARIO

- **CTeI**: Ciencia, Tecnología e Innovación tecnológica.
- **DGAI**: Dirección General de Apoyo a la Investigación.
- **DGFCYT**: Dirección General de Formación en Ciencia y Tecnología.
- **SINACYT:** Sistema Nacional de Ciencia, Tecnología e Innovación Tecnológica.
- **Widget**: también conocido como artilugio o control es un componente gráfico, o control, con el cual el usuario interactúa, como por ejemplo, una ventana, una barra de tareas o una caja de texto.
- **EJB**: La tecnología Enterprise JavaBean (EJB) proporciona la posibilidad de usar componentes software transaccionales que residen en el servidor de aplicaciones. Los componentes software (componentes EJB) pueden ser usados desde cualquier programa Java de forma distribuida.
- **MVC** : Modelo Vista Controlador es un patrón de arquitectura de software que separa los datos de una aplicación, la interfaz de usuario, y la lógica de control en tres componentes distintos. El patrón MVC se ve frecuentemente en aplicaciones web, donde la vista es la página HTML y el código que provee de datos dinámicos a la página.

- **PDA** : del inglés ***P**ersonal **D**igital **A**ssistant*, (Ayudante personal digital) es un computador de mano originalmente diseñado como agenda electrónica. Hoy en día se puede usar como una computadora doméstica (ver películas, crear documentos, navegar por Internet...).
- **NCSA:** acrónimo del *National Certer for Supercomputing Applications* (Centro Nacional de Aplicaciones de Supercomputación). Es un organismo estadounidense relacionado con la investigación en el campo de la Informática y las Telecomunicaciones. Jugó un papel muy importante en el desarrollo del World Wide Web dado que introdujo el visualizador Mosaic.
- **API**: La Interfaz de Programación de Aplicaciones (o su acrónimo en inglés, API - Application Programming Interface) es un conjunto de funciones residentes en bibliotecas (generalmente dinámicas) que permiten que una aplicación corra bajo el sistema operativo.

INDICE DE CUADROS Y FIGURAS